Concept Drift Detection Methods For Deep Learning Cognitive Radios: A Hardware Perspective

Shahriar Shahabuddin^{*†}, Zaheer Khan[†], Markku Juntti[†]

^{*}Mobile Networks, Nokia, Oulu, Finland

[†]Centre for Wireless Communications, University of Oulu, Finland

Email: ^{*}[first.lastname]@nokia.com, [†][firstname.lastname]@oulu.fi

Abstract—Deep learning models usually assume that training dataset and target data have the same distribution. If this is not the case, model mismatch causes performance degradation when the model is used with the real data. With radio frequency (RF) measurements from real data traffic, the exact distribution of the measurements is unknown in many cases and model mismatch is unavoidable. This is known as concept drift, or model misspecification in deep learning, which we are interested in for cognitive radio dynamic spectrum access predictions. In this paper, we present three concept drift detection methods and their corresponding very large scale integration (VLSI) circuits. The circuits are mapped on a Xilinx Virtex-7 field-programmable gate array (FPGA) and the resource utilization results are provided.

Index Terms—cognitive radio, deep learning, concept drift, spectrum sensing, spectrum prediction, VLSI, FPGA.

I. INTRODUCTION

The rapid development of wireless technologies created a scarcity of spectrum resources. However, the utilization rate of spectrum resources is mostly low according to U.S. Federal Communication Commission (FCC). Cognitive radio technology alleviates the spectrum scarcity bottleneck by utilizing the available spectrum opportunistically. Two important techniques for cognitive radio networks are: (1) spectrum sensing and (2) spectrum prediction. The secondary users (SU) are able to distinguish the availability of the shared spectrum through spectrum sensing. The SUs are able to foresee active status of the primary users (PU) and schedule their access in advance with spectrum prediction [1]. The spectrum prediction step generates an initial guess on the availability of an idle channel by a training process based on historical spectrum sensing data. Compared to a generic cognitive radio system, spectrum prediction guarantees more accurate PU and SU scheduling. Thus, spectrum prediction can reduce the time scheduling of spectrum access and increase the overall throughput. Deep neural networks (DNN) have been successfully used for spectrum prediction [2]. The DNNs mostly use offline training under the assumption that the activity status of the PU and SU are relatively stable and the parameters of the prediction model can be used for a long time.

However, radio frequency (RF) measurements from real data traffic indicates that the exact distribution of traffic is sometimes unknown and model mismatch is unavoidable [3]. The model mismatch is commonly known as concept drift or dataset shift in the literature [4]. Concept drift detection

Fig. 1: Intuitive block diagram of a deep learning cognitive radio with concept drift detection. DNN is partitioned between local computing platform and remote server.

method is typically used to describe the unforeseeable changes in the underlying distribution of streaming data over time. As DNN training is a complex and time consuming process, it is crucial to detect and understand the concept drift of streaming data. Concept drift detection has been widely used for DNN solutions of pattern recognition and data stream mining problems. The application of concept drift detection methods for deep learning cognitive radios is still at an early phase.

An intuitive block diagram of a deep learning cognitive radio with concept drift detection is presented in Fig. 1. DNN is partitioned between a remote server and a local computing platform, for example, a field-programmable gate array (FPGA), due to the prohibitive complexity and latency associated with DNN training. In Fig. 1, the concept drift block compares the present and past data distributions and generates an alarm for the DNN training block whenever necessary. Therefore, the samples, which are processed by a digital front end (DFE), are processed by both the baseband processing chain and the concept drift detection block. A concept drift implementation with an FPGA provides twofold advantages: (1) high throughput real time operations are possible, (2) data transfer to the remote server, which is a costly operation, can be minimized.

In this paper, we present FPGA implementation of three concept drift methods based on histogram dissimilarity measures. The methods are: (1) χ^2 divergence, (2) Kolmogorov-

Smirnov distance, (3) match distance. We present a framework for calculating the concept drift and perform word length analysis. Afterwards, we present VLSI architectures of concept drift detection methods, which also include histogram computation. The rest of the paper is organized in the following way: In Section II, we discuss our framework for concept drift detection and present several histogram dissimilarity measures. In Section III, VLSI architectures of the drift detection methods are discussed. FPGA implementation results are presented in Section IV and the conclusions are drawn in Section V.

II. CONCEPT DRIFT DETECTION: HISTOGRAM DISSIMILARITY MEASURES

We assume a system model where the baseband unit at a wireless receiver is receiving I/Q samples from the digital front-end (DFE). A received frame of I/Q samples is divided in N blocks, where the current block is compared to a past block of the same size to calculate the concept drift. A widely used technique to measure concept drift is to find the distance between histograms. We apply the same idea for RF data in our system model. After computing histograms from past and present blocks, the dissimilarity between them is computed to understand the margin of the drift. This framework for calculating concept drift from the received data stream is illustrated in Fig 2.

A. Histogram computation

A histogram $U = \{u_i\}$ is a mapping from a set of d -dimensional integer vectors $\mathbf{i}$ to the set of non-negative reals. The integer vectors represent bins and the non-negative reals represent the occurrence frequency of that bin, which can be as an estimator of the probability mass function (PMF) [5]. We assume an evenly spaced bin width and a fixed number of bins. Once the bin size is selected, different threshold values can be selected for each bin. Whenever a corresponding value arrives, the value of the bin is incremented [3]. The histograms are typically normalized by dividing the value of a bin with the total value of bins in the entire histogram.

B. Histogram dissimilarity measures

There are many methods to measure dissimilarity between two histograms, $U = \{u_i\}$ and $W = \{w_i\}$. They can be divided in two categories: (1) bin-by-bin dissimilarities, (2) cross-bin dissimilarity. Bin-by-bin dissimilarity measures compares contents of corresponding bins between two histograms. The cross-bin measures also considers neighbourhood bins to be treated. The popular cross-bin measures, such as, earth movers distance (EMD), are applicable to two-dimensional (2D) image data. For RF data, we only consider bin-by-bin dissimilarity measures to compute the histogram distance. Numerous bin-by-bin dissimilarity measures exist in the literature [6]. We select three popular ones for further consideration.

1) χ^2 divergence: χ^2 divergence is a widely used dissimilarity measure for histograms based on statistics. There are several χ^2 divergence methods [6]. We use a symmetric χ^2

Fig. 2: Framework for calculating concept drift.

method, which is also known as squared χ^2 distance. The squared χ^2 distance can be expressed as

$$d_{\chi^2}(U, W) = \sum_i \frac{(u_i - m_i)^2}{m_i}, \quad (1)$$

where $m_i = \frac{u_i + w_i}{2}$.

2) *Kolmogorov-Smirnov distance*: Kolmogorov-Smirnov finds the distance between two one-dimensional histograms by calculating the maximum value of the difference between their corresponding cumulative histograms. The Kolmogorov-Smirnov distance can be expressed as

$$d_{KS}(U, W) = \max_i (|\hat{u}_i - \hat{w}_i|). \quad (2)$$

where $\hat{u}_i = \sum_{j \leq i} u_j$ is the cumulative histogram of U .

3) *Match distance*: Match distance calculates the L_1 distance between two cumulative histogram. This is a special case of EMDs for one-dimensional histograms. The match distance can be expressed as

$$d_{MD}(U, W) = \sum_i |\hat{u}_i - \hat{w}_i|. \quad (3)$$

III. VLSI ARCHITECTURE

A. Word length analysis

To perform word length analysis, we assume a fast Fourier transform (FFT) based transceiver. A 15 kHz subcarrier spacing and an FFT size of 1024 is applied. A histogram dissimilarity is calculated for every 1024 samples. Firstly, we apply the same data set for both u_i and w_i . Afterwards, we apply an abrupt concept drift by changing the data distribution of w_i . Finally, we apply fixed-point data set and simulate in Matlab to find the appropriate word length. An example of such a simulation for χ^2 divergence is presented in Fig. 3. For readability, We show 20 blocks in this simulation, where each block consists of 1024 samples. For first ten blocks, we apply the same distribution for both u_i and w_i , and therefore,

Fig. 3: Word length analysis for χ^2 divergence calculation.

the distance between them is zero. From block 11, we apply a different distribution for w_i . Hence, we see the distance between the histograms increased, and not zero anymore. We apply fixed-point data types with an acceptable margin of quantization for hardware implementation, which can be also seen in Fig. 3. The word length analysis of Kolmogorov-Smirnov and match distance are also done in a similar fashion.

B. Histogram computation unit

We generate histograms for every 1024 samples for u_i and w_i . These samples are stored in a first-in first-out (FIFO) read access memory (RAM), which is not part of our design. We assume a fixed number of bins (21) and bin width for a histogram. The histogram computation units starts with a demultiplexer, which forwards the incoming samples to its corresponding bin. We use 21 parallel counters for 21 bins, where each counter can count up to 1024, and thus requires 10 bits. Whenever demultiplexer points to a bin, the counter associated to the bin increases. A separate master counter is used to indicate when a complete set of 1024 samples are demultiplexed. The final step of the histogram computation is normalizing the bin counts with a summation of all bin counts. As we generate a histogram for every 2^{10} samples, a right shift by 10, is sufficient for the normalization operation. To make the processing simpler, we assume the radix point has been shifted to left by 10 bits. With the normalization, we also ensure that the histogram bin count values lies within $[0, 1]$. The values of different bins are transferred to the output sequentially with the aid of another counter.

C. Kolmogorov-Smirnov distance computation

The first step of Kolmogorov-Smirnov distance calculation is computing cumulative histogram. As the normalized histogram bin counts are received sequentially, we apply an accumulator unit (AU) to compute a cumulative histogram. An AU consists of a recursive addition operation as shown Fig. 4. The recursive addition includes an adder, a multiplexer and a register. The multiplexer selects between the first input u_i and addition of u_i and past accumulated value, which is stored in the register. As the addition of the normalized bin count does not exceed 1, we set both the adder and the accumulator register to 10 bits. A counter is applied to control the number of additions to be performed by AU. Similarly, another AU is used to compute the cumulative histogram of w_i . Afterwards, the cumulative histograms are subtracted as shown in Fig. 4. The final step of the Kolmogorov-Smirnov

Fig. 4: VLSI architecture of Kolmogorov-Smirnov distance computation.

calculation is finding the maximum of the subtraction. We apply a vector max (VM) unit, which compares the incoming value with a past stored value and update the register with the maximum value at each cycle. The comparator and the storing register of the VM unit is shown on the right side in Fig. 4.

D. Match distance computation

Match distance VLSI architecture is very similar to that of the Kolmogorov-Smirnov architecture in Fig. 4. From (3), match distance computes an accumulation instead of finding the maximum value of a vector. Therefore, we replace the VM unit of Fig. 4 with another AU for match distance. Hence, another counter is created to control the final AU. Similar to Kolmogorov-Smirnov, all adders and registers for the match distance unit are 10 bits wide.

E. χ^2 divergence computation

χ^2 divergence computation starts with the computation of $m_i = \frac{u_i + w_i}{2}$. The m_i computation consists of an addition of u_i and w_i , followed by a right shift operation to compute the division by 2. From (1), a square operation is required for χ^2 divergence, which we compute with a 18 bit real multiplier. Afterwards, a division operation is required for χ^2 divergence

Fig. 5: VLSI architecture of χ^2 statistics computation.

computation. As a straightforward division operation is challenging for the VLSI, we compute a reciprocal of m at first. The reciprocal can be multiplied with the numerator to obtain the final division output. The inverse $1/m$ can be computed in an iterative manner with Newton-Raphson method as

$$m_{k+1} = 2m_k - mm_k^2, \quad (4)$$

where k is the number of iterations. The initial value m_0 follows the convergence criterion $0 < m_0 < 2/m$. We store the initial values in a look-up table (LUT). To control the dynamic range, a shift operation is applied on m . This shift operation ensures that the result lies in the range of $[1/2, 1]$. Two similar structures for Newton-Raphson division is presented in [7]. We invite interested readers to browse through [7] to gain more insight on such architectures. An 18 bit multiplier is used to obtain the final result for the division. An AU is used to perform the summation of (1). The entire operation of χ^2 divergence circuit is shown in Fig. 5.

IV. FPGA IMPLEMENTATION

Three circuits are developed for Kolmogorov-Smirnov, match distance and χ^2 divergence. In addition to the histogram dissimilarity logic, two histogram computation units are included in each concept drift circuit. The circuits are developed with VHDL on register-transfer level (RTL) and mapped on a Xilinx Virtex-7 XC7VX690T FPGA. For synthesis and implementation strategy, Vivado default settings is used. Xilinx design constraints (XDC) are set to a common 200 MHz clock frequency for all designs. The utilization report of the circuits are presented in Table I.

TABLE I: Resource utilization of concept drift methods

Method	Kolmogorov-Smirnov	match distance	χ^2 divergence
LUT slices	955 (0.22%)	953 (0.22%)	1004 (0.23%)
FF slices	552 (0.06%)	562 (0.06%)	517 (0.06%)
DSP	0	0	4 (0.11%)

It can be seen from Table I that the size of the circuits are relatively small compared to the total resource available in a Virtex-7 FPGA. This is a desirable result as the concept drift detection method will be part of a large system, similar to the one presented in Fig. 1. The Kolmogorov-Smirnov and match distance provide very similar results. This is due to the fact that the circuits are almost identical except the final part, where match distance use a AU instead of a VM unit. The χ^2 use similar number of look-up table (LUT) and flip-flop (FF) slices, however, it uses four DSP units. Two DSP units corresponds to the two real multipliers in the datapath, which are shown in Fig. 5. The other two DSP units correspond to the multiplications required for the reciprocal operation of 4. Although the LUT and FF consumption of χ^2 divergence is also similar, the multiplication units will impact the overall area in an application-specific integrated circuit (ASIC) implementation.

Most FPGA implementations of histograms, which can be found in the literature, pertains to image processing. For example, FPGA implementations of histograms for image processing can be found in [8]–[10]. Xilinx presented a basic histogram computation method in [8], where a 256 bins are implemented for with a block RAM (BRAM) for a 8-bit-per-pixel image. In order to speed up the histogram calculation, in-module parallelism is introduced in [10]. A collision free memory access is introduced in [9]. We apply a counter based implementation as the number of bins in this work is

TABLE II: Comparison of FPGA implementations for histograms and dissimilarity measures

Application	FPGA	LUT	FF	DSP	BRAM
Histogram [8]	Virtex-6	-	8	0	1
Histogram [9]	Virtex-2	217	122	0	2
Histogram [10]	Virtex-2	94	64	0	2
Kolmogorov [11]	Spartan 3A	688	-	0	2
χ^2 [12]	Virtex-2	1681	3233	64	0
Histogram and Kolmogorov	Virtex-7	955	552	0	0
Histogram and χ^2	Virtex-7	1004	562	4	0

smaller. For a large number of bins, a BRAM-based solution is more appropriate. The concept drift detection methods are not a part of [8]–[10]. Histogram dissimilarity measures can be found in the literature as part of energy detection mechanism for cognitive radios [11], [12]. For example, FPGA implementation of a Kolmogorov-Smirnov test statistic based energy detector is presented in [11]. The circuit is implemented on a Spartan 3A FPGA, which is part of a universal software radio peripheral (USRP) device. A χ^2 statistics based energy detector for cognitive radios is presented in [12]. The energy detector has been designed with Xilinx System Generator and mapped on to a Virtex-2 FPGA.

Our implementation is compared to [8]–[12] in Table II. The comparison is not entirely fair as the FPGA devices and the operating frequency are different among the applications. The number of FF slices in [8] is significantly lower than that in any other implementations, however, the number of LUTs are not reported. The numbers of LUTs and FFs of [9] and [10] are $5\times$ to $10\times$ lower than it is in our implementation. However, this can be attributed to the dual histogram in our implementations and the concept drift architecture. The Kolmogorov architecture of [11] takes less LUTs than our implementation and the FF results are not provided. The χ^2 architecture of [12] takes $1.5\times$ more LUT and $3\times$ more FF than our implementation. To our best knowledge, this is the only work that includes VLSI architecture and FPGA implementation of both histogram computation and concept drift for cognitive radios. FPGA implementation of histogram and a concept drift method is previously presented in [3] with the aid of high level synthesis, and thus, the detail of VLSI architecture is missing in [3].

V. CONCLUSIONS

We presented VLSI architecture for three concept drift detection methods. We derived a framework to compute the concept drift from the streaming data and conducted the fixed-point analysis on Matlab. The VLSI architectures are mapped to Xilinx Virtex-7 FPGAs and the resource utilization result is provided. A comparison to a corresponding earlier implementations is also provided. Our hardware design results provide important insight into successful implementation of deep learning cognitive radios.

REFERENCES

- [1] P. Zuo, X. Wang, W. Linghu, R. Sun, T. Peng, and W. Wang, "Prediction-based spectrum access optimization in cognitive radio networks," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 2018, pp. 1–7.
- [2] L. Yu, J. Chen, G. Ding, Y. Tu, J. Yang, and J. Sun, "Spectrum prediction based on Taguchi method in deep learning with long short-term memory," *IEEE Access*, vol. 6, pp. 45 923–45 933, 2018.
- [3] Z. Khan, J. Lehtomäki, C. Ganewattha, and S. Shahabuddin, "Histograms to quantify dataset shift for spectrum data analytics: A SoC based device perspective," in *2020 2nd 6G Wireless Summit (6G SUMMIT)*. IEEE, 2020, pp. 1–5.
- [4] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.
- [5] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International journal of computer vision*, vol. 40, no. 2, pp. 99–121, 2000.
- [6] S.-H. Cha, "Taxonomy of nominal type histogram distance measures," in *Proceedings of the American Conference on Applied Mathematics*, ser. MATH'08. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2008, p. 325–330.
- [7] C. Studer, S. Fateh, and D. Seethaler, "ASIC implementation of soft-input soft-output MIMO detection using MMSE parallel interference cancellation," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 7, pp. 1754–1765, 2011.
- [8] E. Garcia, "Implementing a histogram for image processing applications," *Xcell Journal*, vol. 38, pp. 46–47, 2000.
- [9] A. Shahbahrami, J. Y. Hur, B. Juurlink, and S. Wong, "FPGA implementation of parallel histogram computation," in *2nd HiPEAC Workshop on Reconfigurable Computing*. Published, 2008, pp. 63–72.
- [10] E. Jamro, M. Wielgosz, and K. Wiatr, "FPGA implementation of strongly parallel histogram equalization," in *IEEE Design and Diagnostics of Electronic Circuits and Systems*. IEEE, 2007, pp. 1–6.
- [11] R. Marsalek, M. Pospisil, T. Fryza, and M. Simandl, "Sequential and Parallelized FPGA implementation of spectrum sensing detector based on Kolmogorov-Smirnov test," in *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 2013, pp. 336–345.
- [12] S. Srinu and S. L. Sabat, "FPGA implementation of spectrum sensing based on energy detection for Cognitive Radio," in *International Conference on Communication, Control, and Computing Technologies*, 2010, pp. 126–131.