A MapReduce Framework to Improve Template Matching Uncertainty

Nicholas Napoli*, Kevin Leach†‡, Laura Barnes*, Westley Weimer†
*Department of Systems and Information Engineering, University of Virginia, Charlottesville, VA 22903
†Department of Computer Science, University of Virginia, Charlottesville, VA 22903
‡Charles L. Brown Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA 22903
{njn5fg, kjl2y, lbarnes, weimer} @ virginia.edu

***Abstract*—Normalized cross-correlation template matching is used as a detection method in many scientific domains. To be practical, template matching must scale to large datasets while handling ambiguity, uncertainty, and noisy data. We propose a novel approach based on Dempster-Shafer (DS) Theory and MapReduce parallelism. DS Theory addresses conflicts between data sources, noisy data, and uncertainty, but is traditionally serial. However, we use the commutative and associative nature of Dempster's Combination Rule to perform a parallel computation of DS masses and a logarithmic hierarchical fusion of these DS masses. This parallelism is particularly important because additional data sources allow DS-based template matching to maintain accuracy and refine uncertainty in the face of noisy data. We validate the parallelism, accuracy, and uncertainty of our implementation as a function of the size and noise of the input dataset, finding that it scales linearly and can retain accuracy and improve uncertainty in the face of noise for large datasets.**

I. Introduction

Template Matching methods are ubiquitous in the scientific community. Matching approaches have been implemented across many scientific disciplines ranging from electrocardiogram analysis in medicine [17], [36], to gene expression in biology [30], to signature detection in communications [3], to analysis of seismic signals in geo-science [2], [4], and to image tracking and recognition in computer vision [5], [18], [39]. Traditional template matching methods are popular because they are effective, easily interpreted, and can be evaluated or prototyped rapidly. The fundamental concept behind template matching is to quantify similarity between two *objects*, which are single- or multi-dimensional signals such as time series information or images. There are several ways to quantify similarity, including sum of absolute difference, histogram matching, and selective correlation coefficients [18], [23], [39]. This paper focuses on the popular method of correlation to quantify object similarity. This statistical approach measures the linear relationship between two objects by providing a correlation coefficient as a metric indicating evidence of similarity.

Prior Work. Traditional single-template cross-correlation matching techniques establish the statistical linear relationship between two objects' unique features [26]. A template is serially translated over a signal or image to search for the strongest correlation coefficient [12], [15], [17], and if a threshold criterion is exceeded a match is concluded. Contemporary approaches also consider sets of templates. A template set acts as a source of evidence for detecting various features within a single signal of interest [36]. Each template in the set represents a specific hypothesis; the template with the highest coefficient is reported as the winner. This work raises three interesting questions: 1) what can be done if this serial algorithm does not scale to larger datasets, 2) what should be returned if multiple templates report similar correlation coefficients, and 3) what is the certainty that the reported template is the true match?

Challenges. This issue of ambiguous winners creates an uncertainty regarding which template is the true match [28]. The complexity of the decision is increased when additional sources of information are provided as sets of multiple templates. Current approaches employ voting and weighting strategies [13], [17], [24]. However, these approaches do not appropriately capture the uncertainty associated with deciding a winner [28]. This situation is further compounded when working with high-volume datasets that require significant computation time. Additionally, the veracity of real-world data is frequently degraded by noise and measurement error. We desire a scalable template matching approach that handles multiple templates and a high volume of noisy source data while still capturing ambiguity.

Insights. As relevant datasets become larger and more readily available, there is evidence that increasing the size of the dataset can increase detection specificity and sensitivity [40]. Such benefits are contingent on proper regularization and treatment of noisy samples [40]. Dempster-Shafer (DS) theory offers *combination rules* that can properly account for the regularization of noise [6]. These combination paradigms allow fusion of evidence sources into a single set of hypotheses. Fusing sources allows contextual considerations to be captured, such as conflicts between sources, corrupt information, uncertainty, source reliability, and accuracy [31]. We propose to use a DS theory-based approach for template matching to address the issues of multiple templates and ambiguity; however, the benefits of DS theory cannot be fully leveraged unless the approach applies efficiently to many templates and sources.

The MapReduce distributed programming paradigm has become increasingly well-supported by companies and computing clusters [10]. MapReduce takes advantage of insights from functional and parallel programming to gain high performance, but requires that computations be structured and data be staged with Map and Reduce tasks operating over $\langle key, value \rangle$ pairs. While MapReduce has been used successfully in research [21], [25] and industrial practice [10], [32], to our knowledge it has

not been used to support DS theory. This may result from the relative nascency of MapReduce and the variety and novelty of DS theory frameworks to form evidence. We exploit the associativity and commutativity of Dempster's Combination Rule to produce a parallel Map and a hierarchical logarithmic Reduce over $\langle key, value \rangle$ pairs representing DS correlation information, to scale template matching to large datasets.

Contributions. We design and implement a parallelized and distributed framework based on the MapReduce paradigm that carries out Dempster-Shafer theory calculations to perform multiple-set template matching in a manner that handles large datasets, ambiguity, and noisy data.

The contributions of this work are:

1) We develop a MapReduce framework for an evidence fusion methodology that can leverage large volumes of templates and sources to improve uncertainty.
2) We demonstrate that our framework scales to high volumes of data.
3) We demonstrate that our framework is robust against noisy data.

II. Background

In this section, we introduce the MapReduce framework for distributed computation and the Dempster-Shafer Theory approach evidence fusion.

A. MapReduce

MapReduce [10] is a programming paradigm intended for distributing computations over large datasets on a cluster. In principle, MapReduce consists of two phases: *Map* and *Reduce*, named after the map and reduce (or fold) functions in functional programming. MapReduce is powerful in situations where a large amount of input can be processed independently (e.g., embarrassingly parallel applications).

The Map phase transforms, filters or sorts data in parallel. Map operates on each element of the input (represented as a $\langle key, value \rangle$ pair) and produces zero or more $\langle key, value \rangle$ pairs as output. Map should be stateless, operating only on its input. Many instances of the Map function can execute simultaneously (e.g., on different nodes) because there are no dependencies between pairs.

After the Map phase, the output may be staged or exchanged between nodes, and $\langle key, value \rangle$ pairs are sorted and assigned (or partitioned) to nodes for reduction. The Reduce function is applied once for each key, accessing all of the $value$s associated with that key and producing zero or more outputs.

MapReduce frameworks or implementations handle staging, marshaling, and data transfer aspects while the user provides a few specified functions (e.g., reading the input files, Map, Reduce, writing the output, etc.). In this paper, we take advantage of the commutativity and associativity of Dempster's Combination Rule to realize a highly parallelized framework for DS fusion. Traditionally, MapReduce parallelism benefits come from the Map stage; hierarchical reductions such as the one proposed in Section III-D are not part of most MapReduce frameworks.

B. Dempster-Shafer Theory Background and Notation

Dempster-Shafer Theory is an evidence-based approach which develops support for hypotheses. Evidence is constructed around information regarding occurrences of events [16]. A single piece of evidence may support multiple hypotheses. To create the necessary evidence, information is gather from *sources*, which can be sensors, organizations, databases, people, or other entities [16].

DS Theory is a generalization of classical probability theory [34], where the support of hypotheses can be considered propositions. These propositions, referred to as the *frame of discernment* (FOD), are mutually exclusive and exhaustive. We take the FOD Ω to be a finite set (i.e., $\Omega = \{\theta_1, \ldots, \theta_N\}$). Ω is finite and composed of N *singleton* propositions. The *basic probability assignment* (BPA), otherwise referred to as a *basic belief assignment* or *mass function*, is a function $m : 2^\Omega \to [0, 1]$, where 2^Ω is the power set of Ω, such that all probabilities sum to 1:

$$m(\emptyset) = 0; \quad \sum_{X_i \subseteq 2^\Omega} m(X_i) = 1. \tag{1}$$

While $m(X)$ measures *only* the support that is directly assigned to proposition $X \subseteq \Omega$, the *belief* $Bl(X)$ represents the total support that can move into X from all other propositions that contain X. Thus, $Bl(X) = \sum_{Y \subseteq X} m(Y)$. Belief is the minimum amount of support that is given for a specific proposition. For the singleton case, the DS mass of the proposition is equal to the belief.

Capturing Uncertainty via DS Theory. Uncertainty has many ways of entering a model. There are two classifications of uncertainty: *aleatory uncertainty* and *epistemic uncertainty*. Aleatory uncertainty represents unknowns that differ each time the system is observed, and can be accounted for using historic data. In this work, we focus on epistemic uncertainty, caused by a lack of knowledge, which is reduced through increased understanding [31], [34]. Hence, when more evidence is provided a refinement of the uncertainty for the decision is possible through increased knowledge.

DS Theory is a major tool for deciphering uncertainty. It uses data fusion techniques to reduce uncertainty from imperfect data (e.g., source information that is conflicting or sources reporting similar information) [22], [31], [38]. Combining additional sources of evidence, referred to as *evidence fusion* [27], reduces uncertainty and redistributes masses to the propositions. Therefore, when one source of evidence indicates a specific template and another source conflicts with that prediction, the uncertainty in the model increases. Likewise, if there is no conflict and both sources report similar findings, the uncertainty decreases. The evidence fusion approach implemented in this paper uses *Dempster's Combination Rule* (DCR):

$$m(X_i) = \frac{\sum_{X_p \cap X_q = X_i} m_1(X_p)\, m_2(X_q)}{1 - \sum_{X_p \cap X_q = \emptyset} m_1(X_p)\, m_2(X_q)}, \tag{2}$$

where the evidence provided by the mass functions m_1 and m_2 are combined to obtain the fused mass function m. We write

$m = m_1 \oplus m_2$ to denote that m is the fused mass function produced by combining m_1 and m_2.

Finally, we note that the DCR $\oplus$ has two important mathematical properties: it is commutative (i.e., $m_1 \oplus m_2 = m_2 \oplus m_1$) and associative (i.e., $(m_1 \oplus m_2) \oplus m_3 = m_1 \oplus (m_2 \oplus m_3)$). While these properties of DCR are well-known in the literature [16], [31], to our knowledge they have not been previously used to support a distributed implementation of DS Theory.

III. MapReduce Framework for Template Fusion

To position our framework within a MapReduce environment, we describe two algorithms, `Map` and `TreeReduce`, that compute a winning match. Typical implementations compute a match serially. However, by noting the associativity and commutativity of Dempster's Combination Rule, we can execute our Map and TreeReduce functions in parallel, gaining significant speedups when a computing cluster is available. In addition, the structure of our inputs is pivotal to the correct execution of these algorithms.

A. Source Dataset

The input dataset is structured to properly combine MapReduce with DS Theory. This dataset contains D different sources, with an individual source denoted by $\mathbf{S}_d$. Each source is considered to provide evidence for its own FOD, and all sources contain the same number of templates, N, in the same order. A specific template i associated with source d is denoted as $T_{d,i}$.

Each template i supports a specific hypothesis. As each source contains a series of templates in the same order, each source can thus be thought of a series of hypotheses. For example, template $T_{1,i}$ in source $\mathbf{S}_1$ reflects the same proposition/hypothesis generated by template $T_{2,i}$. Furthermore, every template in every source is identical in size.[1] This is a necessary precondition for the computation of correlation coefficients. In practice, templates can be uniformly resized in a preprocessing step.

Each set of templates is examined to create our support for each proposition in the set. This structure allows each source to be reconstructed into its own FOD (Ω), which is crucial for MapReduce because a single FOD can be represented by a single $\langle key, value \rangle$ pair.

B. Correlation to DS Framework: Map

The Map function's task is to analyze a source's information and develop evidence using correlation. This set of correlation coefficients is then placed into a DS Framework [28]. There are two inputs to the Map function. The first input is an unknown 2-dimensional signal, A, with dimensions m by n. The second input is a Source, $\mathbf{S}_d$, composed of a set of N templates, where each template's dimension are m by n.

[1] In our evaluation, we investigate an example image processing task in which we consider every image to have the same dimensions.

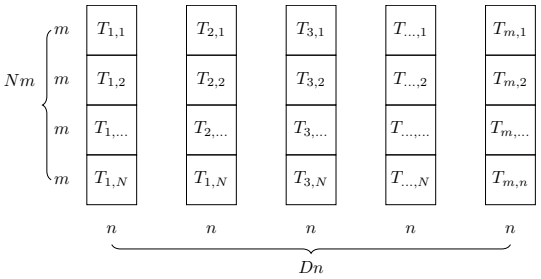


Fig. 1. Visual representation of the structure of input sources. Each template is an $m \times n$ signal, each source contains N different templates. There are D sources in the dataset.

1) Forming Evidence Using Correlation: To develop evidence, correlation between the unknown template and a known template, $T_{d,i}$, produces a correlation coefficient. This correlation coefficient is assigned to the template, providing evidence of the match. A cross-correlation value of $\rho = 1$ denotes a perfect match between the unknown and known templates. When $\rho = 0$, there is no match, and when $\rho = -1$, the two signals are out of phase or negatively correlated. Since we are examining a set of correlation coefficients from a source, this framework can be applied to multidimensional template matching. In this paper, our work considers the two dimensional case (e.g., images), where the correlation coefficient can be expressed as

$$\rho_{\mathbf{A},\mathbf{B}} = \frac{\sum_{j=1}^{m}\sum_{i=1}^{n}(A_{ij} - \overline{\mathbf{A}})(B_{ij} - \overline{\mathbf{B}})}{\sqrt{\left(\sum_{j=1}^{m}\sum_{i=1}^{n}(A_{ij} - \overline{\mathbf{A}})^2\right)\left(\sum_{j=1}^{m}\sum_{i=1}^{n}(B_{ij} - \overline{\mathbf{B}})^2\right)}}, \quad (3)$$

where we denote the 'sample' means of images A and B as

$$\overline{\mathbf{A}} = \frac{1}{mn}\sum_{i=1}^{n}\sum_{j=1}^{m}A_{ij}, \quad \text{and} \quad \overline{\mathbf{B}} = \frac{1}{mn}\sum_{i=1}^{n}\sum_{j=1}^{m}B_{ij}.$$

This process normalizes the correlation coefficient, ρ, where $\rho \in [-1, 1]$. A set of normalized correlation coefficients is generated from $\mathbf{S_d}$'s set of templates by using equation 3 and the unknown template thus take the form

$$\mathbf{V} = \begin{bmatrix} V_1 & V_2 & \cdots & V_N \end{bmatrix}^T, \quad (4)$$

where $V_i \in [0, 1]$ to only denote the positive normalized correlation coefficient. The negative correlation coefficients are set to zero. Further discussion can be found in Napoli *et al.* [28]. Each V_i provides evidence on how similar $T_{d,i}$ is to the unknown template. This correlation calculation is carried out for all sources in a parallelized manner.

C. Set of Correlations Coefficients to DS Masses

We follow the DS framework of Napoli *et al.* [27] for correlation coefficients to capture the overall certainty of each

element in $\mathbf{V}$ by utilizing a weighting strategy as

$$\Delta\mathbf{W} = \Delta\mathbf{V} \circ \mathbf{J}_{NN}\mathbf{D}_N = \begin{pmatrix} V_1\Delta V_{11} & V_2\Delta V_{21} & \cdots & V_N\Delta V_{N1} \\ V_1\Delta V_{12} & V_2\Delta V_{22} & \cdots & V_N\Delta V_{N2} \\ \vdots & \vdots & \ddots & \vdots \\ V_1\Delta V_{1N} & V_2\Delta V_{2N} & \ldots & V_N\Delta V_{NN} \end{pmatrix}, \quad (5)$$

where $\circ$ denotes the matrix Hadamard product, $\Delta V_{ij} = (V_i - V_j) \in [-1, +1]$. $\mathbf{J}_{NM}$ denotes the $N \times M$ matrix with each entry being 1 and $\mathbf{D}_N = diag\,[V_1, V_2 \cdots, V_N]$ denotes the diagonal matrix with the diagonal entries being $\{V_1, V_2, \cdots, V_N\}$. The columns of $\Delta\mathbf{W}$ are references for each V_i proposition being analyzed. The summation of these column vectors informs us the strength of a proposition being a winner relative to other propositions in $\mathbf{V}$. The sum of all the elements in a column vector is referred to as the *column weight*. The *column weights*, C_i, are calculated as

$$\mathbf{C} = [C_1 \quad C_2 \quad \cdots \quad C_N]^T = (\mathbf{J}_{1N}\Delta\mathbf{W})^T, \quad (6)$$

where $C_i \in [-(N-1)/4, (N-1)]$.

We determine P different *focal elements* from the positive values in the column weights vector, $\mathbf{C}$. These focal elements are assigned DS masses. We define a *mass measure vector* $\mathbf{H} = [H_1, H_2, \cdots, H_N]$ as

$$H_i = \frac{C_i + |C_i|}{2}. \quad (7)$$

We first observe that each $H_i \in [0, (N-1)]$. More generally, $\mathbf{H}_d$ is associated with source $\mathbf{S}_d$. Note that, $\mathbf{H}_d$ is associated with source $\mathbf{S}_d$. The mass measure is adjusted to DS masses via

$$M(A) = \begin{cases} 1 - \left(\frac{Y}{(N-1)P}\right), & \text{for } A = \Theta; \\ H_i \left(\frac{1-m(\Theta)}{Y}\right), & \text{for } A = H_i, \end{cases} \quad (8)$$

where $\Theta = \{V_1, V_2, \cdots, V_N\}$ and $Y = \sum_{i=1}^{N} H_i$ is the FOD consisting of the propositions H_i, $i \in 1, \ldots, N$. Once the correlation coefficients for a single source have been converted to masses, those mass vectors can be fused.

D. DCR Fusion: TreeReduce

Our TreeReduce function takes any two FOD vectors V_1 and V_2 as input. The TreeReduce function fuses the two vectors into a single DS mass vector of the same length. The TreeReduce function can then be executed hierarchically to fuse all D FOD vectors in $\log(D)$ steps until one fused vector remains. This final output vector contains the fused DS masses from all the processed sources. From this output, we account for the belief and uncertainty of the winning template based upon the maximal mass in the vector.

Algorithm 1 — Map: calculate DS mass from one source.

Input: A, and unknown input observation to match
Input: B, a single source containing N templates
Input: corr, a subroutine implementing Equation 3
Output: correlation vector OUT of length $N+1$

```
    count = 0
 1: for i = 0 to N − 1 do
 2:     C[i] = corr(A, B[i])
 3: for i = 0 to N − 1 do
 4:     for k = 0 to N − 1 do
 5:         OUT[i] = OUT[i] + (C[i] − C[k])*C[i]
 6:     if OUT[i] < 0 then
 7:         OUT[i] = 0
 8:     else
 9:         count = count + 1
10: sum = sum(OUT)
11: if sum > 0 then
12:     uncertainty = 1 − sum/count
13:     for i = 0 to N − 1 do
14:         OUT[i] = OUT[i] ∗ ((1 − uncertainty)/sum)
15:     OUT[N] = uncertainty
16: else
17:     OUT[N] = 1
```

Algorithm 2 — TreeReduce: Dempster's Combination Rule

Input: V_1 and V_2, mass vectors of length $N+1$
Output: OUT, a fused vector of length $N+1$

```
    tmp = (N + 1) × (N + 1) array
    norm = 0
 1: for i = 0 to N + 1 do
 2:     for k = 0 to N + 1 do
 3:         tmp[i][k] = V1[i] ∗ V2[k]
 4: for i = 0 to N + 1 do
 5:     norm = norm + tmp[i][i] + tmp[i][N] + tmp[N][i]
 6: for i = 0 to N do
 7:     OUT[i] = (tmp[i][i] + tmp[i][N] + tmp[N][i])/norm
 8: OUT[N] = tmp[N][N]/norm
```

IV. Architecture and Implementation

In this section, we describe the architecture and implementation of two prototypes for our parallel framework. The first prototype for running our framework uses Hadoop [1], which allows arbitrary scalability depending on the number of nodes and cores available in a distributed compute cluster. The second prototype is a flexible pthreads implementation in Linux that uses the MapReduce paradigm akin to Phoenix [29]. This enables rapid debugging of our framework on a single node in instances where a Hadoop cluster is not practical.

A. Testing Methodology

We examine changes in mass vectors as a function of source count and compute core count. The system's workflow is illustrated in Figure 2 using a simple handwriting letter classification task. Each source contains an array of template images corresponding to alphanumeric characters. As illustrated in Figure 2, all templates across all sources maintain the same order (described in Section III-A). The parallel Map operation produces an mass vector using the unknown input

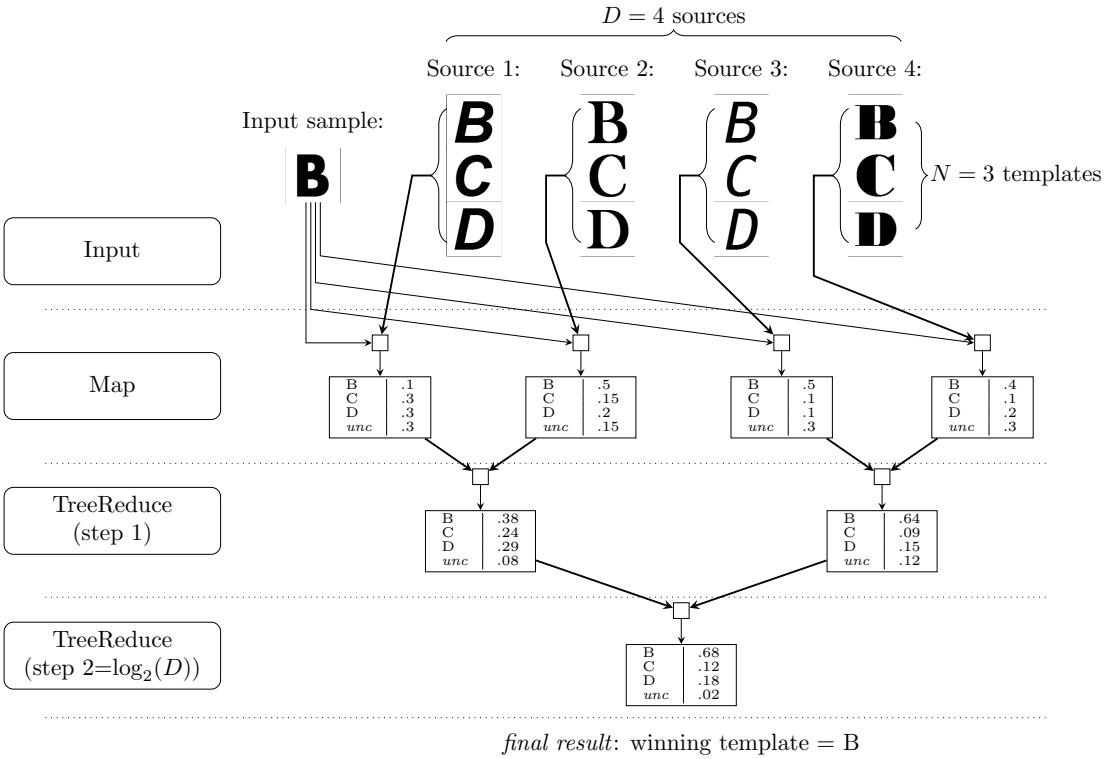


Fig. 2. Architecture of the proposed framework on a handwriting identification task using template matching. D refers to the number of sources present in the system, N refers to the total number of templates each source provides. The Map operation correlates the input image with the ith template image of the dth source ($T_{d,i}$), producing a vector of masses. Pairs of masses are provided to each TreeReduce operation, which fuses the two vectors together. In particular, the fused vector has an increased mass for templates that already had higher masses in the input vector. unc represents the uncertainty.

image and a particular source. The TreeReduce operation then applies DCR in parallel, fusing pairs of mass vectors together and outputting a single updated mass vector. The TreeReduce operation is repeated $\log(D)$ times until a single fused mass vector remains.

B. MapReduce Implementation

Our framework takes advantage of popular open source tools for massively parallel computation. We use Apache Hadoop [1], Hadoop FS [33], and Apache YARN [37] for cluster and computational resource management. These tools apply our framework to any general Hadoop cluster with minimal human effort. In addition, we take advantage of MapReduce4C (MR4C [14]), a C wrapper to run jobs on a Hadoop cluster. MR4C allows executing an arbitrary C function in a Hadoop task managed by YARN. Our overall framework enables the use of all the cores in a cluster to efficiently execute our implementation while reaping the benefits of temporal parallelism.

The first part of the implementation described in Algorithm 1 serves as the Map function. The $\langle key, value \rangle$ pair identifies the source and the array of mass values, respectively. In MR4C, the key would be the name of a source file and the $value$ would be its contents. The output of Map is then a single $\langle key, value \rangle$ pair: the key is unchanged, but the output $value$ is a vector of DS masses produced by the data values from that source and the unknown input sample. Because the inputs are independent, many instances of the Map function can execute simultaneously.

The second part of the implementation described in Algorithm 2 serves as the Reduce function. This function takes two $\langle key, value \rangle$ pairs and applies Dempster's Combination Rule ($\oplus$), producing a single $\langle key, value \rangle$ pair. To further leverage the associative and commutative properties of DCR, a logarithmic TreeReduce is implemented rather than a traditional serial reduction. Ultimately, this improves the total Reduce latency from D to $\log(D)$.

C. Pthreads Implementation

To demonstrate the generality of our framework, we also implement it using pthreads as the underlying parallelism framework. While pthreads cannot directly apply to a distributed environment, it can be used in rapid prototyping situations to take advantage of all of the cores on a machine.

In a parent process, we load the input sample and the sources into global memory (e.g., via `mmap()`). We then create a new thread for each source (and thus each instance of the Map function). Each thread j can independently read from global memory to compute its own mass vector (relating source j to the input sample). We synchronize all threads with a barrier before, we hierarchically execute our TreeReduce function in multiple threads to fuse the mass vectors produced in the Map step. Each level of the TreeReduce hierarchy is similarly synchronized with a barrier before continuing. While threading is limited to shared memory machines, our approach can take advantage of thread pools (e.g., with BoostThreads [9]) in situations where the source count is extremely large.[2]

[2]Linux often soft-caps thread count to about 32,000; having more sources would be inappropriate for pthreads alone.

V. Evaluation and Discussion

In this section, we validate our claims that our distributed MapReduce implementation of template matching using DS theory. This Framework combines the handling of uncertainty using DS Theory with the efficiency and scalability of MapReduce. We seek to answer the following research questions:

- RQ1 Does our framework scale as the number of sources increases and as the number of computational resources increases?
- RQ2 Is our framework robust against noise that corrupts the data sources?
- RQ3 Does increasing the number of sources reduce the amount of ambiguity and uncertainty in our final winning template?

For evaluation purposes, we consider the task of two-dimensional letter classification. In practice, there are many other domain-specific approaches for handwriting recognition (e.g., [8], [11], [19]). We consider this application because it simplifies the generation of additional sources as well as the introduction and interpretation of noise and uncertainty in a way we can control. However, our approach can be applied to other domains as well. So long as the input dataset meets the structure criteria described in Section III-A, DS mass values can be computed for propositions related to that dataset.

We consider two datasets: the Chars74K [35] dataset of computer fonts, and the MNIST [20] handwriting dataset. Chars74K contains images of numbers and characters using about 1000 different fonts—we consider each font to be a different source of information for our purposes. The MNIST handwriting dataset contains handwritten numbers from unique writers—here, each writer is a different source. Varying amounts of salt-and-pepper noise [7] were added to the data source images. The noise is characterized by a noise density (ϵ). An ϵ of 0.5 equates to 50% of pixels randomly assigned either black or white. Figure 5 illustrates how this noise affects the template images. The prediction accuracy is measured as the fraction of trials that yielded a correct prediction over the number of total trials. A correct prediction is a True Positive (TP); an incorrect one is a False Positive (FP).

We discuss and present experimental results that address each of the research questions below. In each experiment, we also verified the results of our parallel or distributed algorithm against a previous independent serial implementation: in each trial our algorithm and the reference provided the same answer.

A. RQ1 — Scalability

We address the scalability question by measuring total latency as a function of both the number of sources and the number of compute nodes available. We vary the number of sources D used when executing our framework, noting the starting and completion times to determine the total amount of time elapsed (with $n = 50$ repeated trials per measurement). We also repeat this experiment by varying the number of compute cores available to the framework. We normalize the results according to the longest time taken during serial operation.

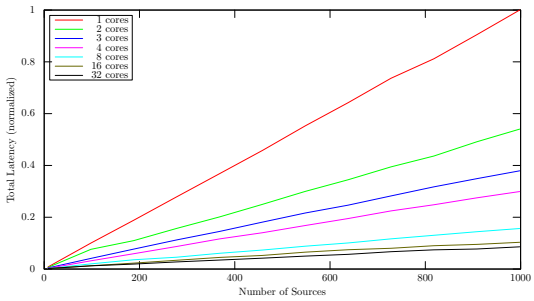


Fig. 3. Total execution latency as a function of source count. The line slopes indicate linear scalability with additional computational resources.

Figure 3 shows the results of this timing experiment for our pthreads implementation. The Hadoop results are similar. Each line corresponds to a different number of cores available to the framework during execution. The lower the slope of the line, the faster it completes the matching task. We note a linear relationship in the number of sources—the total latency is directly related to the number of sources, indicating scalability. Additionally, by taking ratios of the slopes of the lines, we can compute the amount of speedup obtained by doubling the number of available cores. We find that doubling the number of available cores approximately yields a 1.9x speedup. We suspect that switching, staging, and networking overhead contribute to performance degradation at a high number of sources.

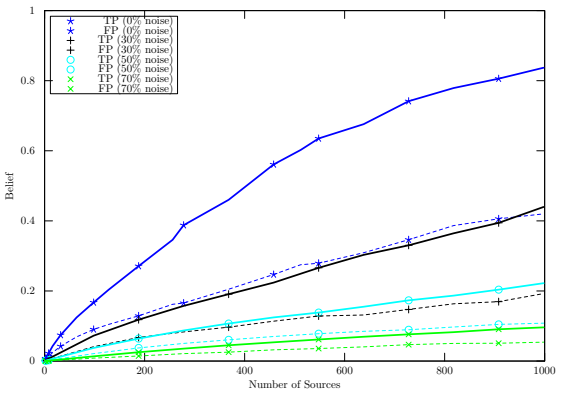
B. RQ2 — Robustness against noise

We hypothesize that having many sources, even in the presence of noise, benefits our DS framework by allowing refinement of the uncertainty and DS masses. We executed our framework, over $n = 1000$ trials, with varying numbers of sources and measured the accuracy of our predictions. Once the trail is complete, we select highest mass to predict in the input image. Figure 6 shows that when we consider a high number of sources (the top two lines), increasing noise does not degrade the accuracy of our framework. These results indicate that our highly-parallel MapReduce framework allows us to leverage a high number of sources to make robust decisions in the presence of large, corrupted information sources.

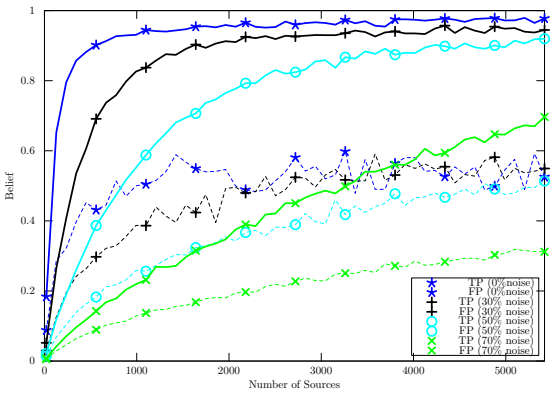
C. RQ3 — Reducing uncertainty with more sources

One key advantage of DS Theory for matching is that each classification answer comes equipped with an uncertainty value for your decision. Scientists can make use of quantified uncertainty to help interpret results: high belief and lower uncertainty in a prediction is valued. We demonstrate that increasing the number of sources results in higher belief and lower uncertainty for predictions from our framework, even in the presence of noise.

Figure 4(a) shows belief as a function of sources and noise in the Chars74K dataset, broken down by FP vs. TP predictions. Consider the lines corresponding to 0% noise: at 200 sources, correct answers come with a belief of about 0.25 while incorrect answers come with a belief of about 0.15: a small margin. By contrast, at 1000 sources, the margin has spread to about 0.8 vs. 0.4. The trends for data with reasonable



(a)



(b)

Fig. 4. Belief vs. number of sources at different noise densities. 4(a) shows results for the Chars74K dataset, while 4(b) shows the MNIST results.

amounts of noise (e.g., 30%) are similar. Figure 4(b) shows the same results using the MNIST handwriting dataset. Again, at 1000 sources, the margin for noiseless predictions 0.9 vs. 0.45.

Additionally, the data suggest that adding many sources in the presence of noise helps the system gain more confidence in its predictions. In other words, our MapReduce framework is capable of lowering uncertainty in the presence of noisy data by leveraging high amounts of noisy sources. In particular, Figure 7 shows the uncertainty for correct and incorrect predictions at varying levels of noise. These results show how increasing the number of sources leads to a more robust prediction.

We note that our current framework considers only singleton cases—that is, single hypotheses. Restricting attention to singletons admits a lower complexity class for the TreeReduce function ($\mathcal{O}(n^2)$a for Algorithm 2). Addressing non-singletons increases the complexity of the TreeReduce function as it requires computing matrix cross-products ($\mathcal{O}(n^3)$). We leave the non-singleton case for future work, but note that it creates further potential for exploiting the massive parallelism we achieved to reduce uncertainty and resolve ambiguities in a practical runtime with a large number of sources.

VI. Conclusion

In this paper, we present a novel MapReduce framework for improving uncertainty in DS template matching. By taking advantage of the associativity and commutativity of Dempster's Combination Rule, we construct a highly scalable framework for fusing evidence in the form of DS mass vectors from many sources of information.

We implement two prototypes of our framework using open source software—one using Hadoop, and the other using pthreads. The pthreads implementation allows rapid development and debugging cycle where testing is required before full-scale deployment. The Hadoop implementation allows rapid deployment since there is very little effort required to deploy a MapReduce algorithm on a Hadoop cluster once written.

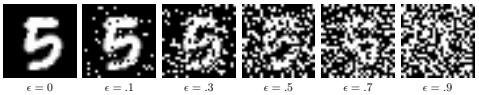


Fig. 5. A template image shown with varying noise densities ϵ which represents the fraction of pixels that are impacted by noise.

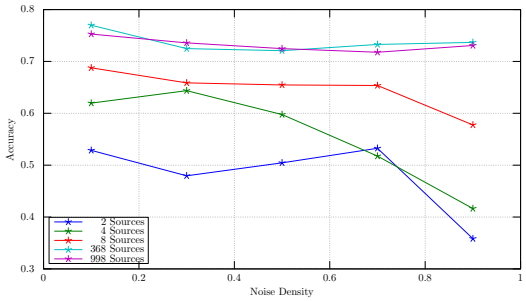


Fig. 6. Prediction accuracy as a function of noise density and source count. The horizontal lines at the top demonstrate that many sources allow for high accuracy even in the presence of noisy data.

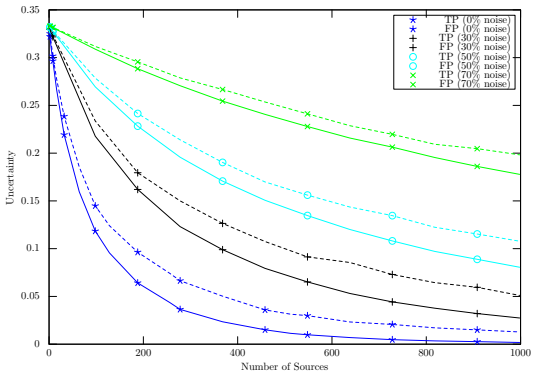


Fig. 7. Uncertainty vs. number of sources at different noise densities in the Chars74K dataset.

We demonstrate the scalability of our system by comparing total execution latency as a function of both source count and the computational elements available. Our system yields about a 1.9x speedup every time the number of cores is doubled. This linear scalability allows practitioners to consider more sources,yielding benefits for accuracy, belief, and uncertainty in the face of noise. We show that providing many sources to our framework allows it to maintain a high level of accuracy even when source data is affected by noise, while maintaining practicality. Finally, we show that our system improves belief

and uncertainty when considering many sources.

ACKNOWLEDGMENT

We acknowledge the partial support of the National Science Foundation (CCF 0954024, CCF 1116289), the US Air Force (FA8750-15-2-0075), the National Institute of Aerospace, and the Virginia Space Grant Consortium. The authors would like to thank Adam Brady of Google for thoughtful discussions of MapReduce in practice.

REFERENCES

[1] "Apache Hadoop," https://hadoop.apache.org.

[2] T. Addair, D. Dodge, W. Walter, and S. Ruppert, "Large-scale seismic signal analysis with hadoop," *Computers and Geosciences*, vol. 66, pp. 145 – 154, 2014.

[3] I. A. Aljarrah, A. S. Ghorab, and I. M. Khater, "Object recognition system using template matching based on signature and principal component analysis," *International Journal of Digital Information and Wireless Communication*, vol. 2, no. 2, pp. 156–163, 2012.

[4] D. Bobrov, I. Kitov, and L. Zerbo, "Perspectives of cross-correlation in seismic monitoring at the international data centre," *Pure Applied Geophys*, vol. 171, p. 439???468, 2014.

[5] K. Briechle and U. D. Hanebeck, "Template matching using fast normalized cross correlation," in *Aerospace/Defense Sensing, Simulation, and Controls*. International Society for Optics and Photonics, 2001, pp. 95–102.

[6] A.-S. Capell, O. Colot, and C. Fernandez-Maloigne, "3d segmentation of mr brain images into white matter, gray matter and cerebro-spinal fluid by means of evidence theory," in *Artifical Intelligence in Medicine 9th Conference of Article Intelligence in Medicine in Europe*, vol. 9. AIME, 2003, p. 5.

[7] R. H. Chan, C.-W. Ho, and M. Nikolova, "Salt-and-pepper noise removal by median-type noise detectors and detail-preserving regularization," *Image Processing, IEEE Transactions on*, vol. 14, no. 10, pp. 1479–1485, 2005.

[8] D. C. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," *CoRR*, vol. abs/1202.2745, 2012.

[9] B. Dawes, D. Abrahams, and R. Rivera, "Boost C++ libraries," http://www.boost.org.

[10] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[11] L. Deng and D. Yu, "Deep convex net: A scalable architecture for speech pattern classification," in *Proceedings of Interspeech*, 2011.

[12] S. Dobbs, N. Schmitt, and H. Ozemek, "QRS detection by template matching using real-time correlation on a microcomputer," *Journal of Clinical Engineering*, vol. 9, pp. 197–212, 1984.

[13] G. Ganeshapillai and J. Guttag, "Real time reconstruction of quasiperiodic multi parameter physiological signals," *EURASIP Journal on Advances in Signal Processing*, vol. 173, no. 1, pp. 1–15, 2012.

[14] Google, "MapReduce4C," https://github.com/google/mr4c.

[15] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*. Addison-Wesley, 1992, vol. 2.

[16] R. U. Kay, "Fundamentals of the Dempster-Shafer theory and its applications to system safety and reliability modelling," *Reliab. Theory Appl*, vol. 3, pp. 173–185, 2007.

[17] V. Krasteva and I. Jekova, "QRS template matching for recognition of ventricular ectopic beats," *Annals of Biomedical Engineering*, vol. 35, pp. 2065–2076, December 2007.

[18] A. Kumar, A. Joshi, A. Kumar, A. Mittal, and D. Gangodkar, "Template matching application in geo-referencing of remote sensing temporal image," *International Journal of Signal Processing, Image Processing and Pattern Recognition*, vol. 7, no. 2, pp. 201–210, 2014.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[20] Y. LeCun, C. Cortes, and C. Burges, "The MNIST Database of Handwritten Digits," http://yann.lecun.com/exdb/mnist/.

[21] S. Lee, S. Jo, and J. Kim, "MRDataCube: Data cube computation using MapReduce," in *International Conference on Big Data and Smart Computing*, 2015, pp. 95–102.

[22] E. Lefevre, O. Colot, and P. Vannoorenberghe, "Belief function combination and conflict management," *Information Fusion*, pp. 149–162, 2002.

[23] Y.-S. Liu, D.-J. Duh, S.-Y. Chen, R.-S. Liu, and J.-W. Hsieh, "Scale and skew-invariant road sign recognition," *International Journal of Imaging Systems and Technology*, vol. 17, pp. 28–39, 2007.

[24] U. Mahbuba, H. Imtiaza, T. Roya, S. Rahmana, and A. R. Ahadb, "A template matching approach of one-shot-learning gesture recognition," *Pattern Recognition Letters*, vol. 34, no. 15, pp. 1780–1788, November 2013.

[25] S. Moon, J.-G. Lee, and M. Kang, "Scalable community detection from networks by computing edge betweenness on MapReduce," in *International Conference on Big Data and Smart Computing*, Jan 2014, pp. 145–148.

[26] D. S. Moore, *The Basic Practice of Statistics*, 3rd ed. W.H Freeman and Company, 2004.

[27] N. J. Napoli, "The detection of analytes using spectroscopy: A Dempster-Shafer approach," Thesis, University of Miami, Coral Gables, FL, August 2014.

[28] N. J. Napoli, L. Barnes, and K. Premaratne, "Correlation coefficients based template matching: Accounting for uncertainty in selecting the winner," in *International Conference on Information Fusion*, July 2015, pp. 1–8.

[29] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis, "Evaluating mapreduce for multi-core and multiprocessor systems," in *High Performance Computer Architecture, 2007. HPCA 2007. IEEE 13th International Symposium on*. Ieee, 2007, pp. 13–24.

[30] I. P. S. Wang and, D. Johnson, F. G. Ibrahim Emam, A. Oehmichen, and Y. Guo, "Optimising parallel r correlation matrix calculations on gene expression data using mapreduce," *BMC Bioinformatics*, vol. 15, no. 351, 2014.

[31] K. Sentz and S. Ferson, "Combination of evidence in Dempster-Shafter theory," Binghamton University, Thesis, 2002.

[32] S. Shankland, "Google spotlights data center inner workings," http://www.cnet.com/news/google-spotlights-data-center-inner-workings/.

[33] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *IEEE Symposium on Mass Storage Systems and Technologies*, 2010, pp. 1–10.

[34] L. P. Swiler, T. L. Paez, and R. L. Mayes, "Epistemic uncertainty quantification tutorial," in *Proceedings of the IMAC-XXVII*, 2009.

[35] T. E. de Campos. (2012, Oct.) The Chars74K dataset: Character recognition in natural images. University of Surrey. Guildford, Surrey, UK. [Online]. Available: http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/

[36] C. N. T. Last and F. Owens, "Multi-component based cross correlation beat detection in electrocardiograms analysis," *Biomedical Engineering Online*, July 2004.

[37] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwieler, "Apache hadoop yarn: Yet another resource negotiator," New York, NY, USA, pp. 5:1–5:16, 2013.

[38] M. Wafa and E. Zagrouba, "Estimation of mass function in evidence theory for fusion of gray level based images," *The International Conference on Signal and Electronic Systems*, September 2010.

[39] J. Yoo and C. Ahn, "Image matching using peak signal-to-noise ratio-based occlusion detection," *IET Image Process*, vol. 6, no. 5, pp. 483–495, 2012.

[40] X. Zhu, C. Vondrick, D. Ramanan, and C. Fowlkes, "Do we need more training data or better models for object detection?." in *BMVC*, vol. 3. Citeseer, 2012, p. 5.