Abdur Rafay Saleem
University of Central Florida
ab464825@ucf.edu

***Abstract*—This is a paper review that introduces a new hashing framework for multimedia retrieval. It is organized in three parts: an explanation of the research problem and technical contributions of the paper, an explanation of the proposed techniques, and a personal critique of the effectiveness of these techniques.**

I. RESEARCH PROBLEM

As the internet is growing day by day, so is the number of users actively using it. More and more users are both producing and consuming large amounts of data. As such, the size of multimedia content, the demand for retrieving and the challenge of serving it fast is growing everyday. Hashing based techniques have been used in the past to reduce this high dimensional data into simpler binary codes which are easier to manage and faster to search for similar content. Learned hashing techniques preserve the metrics in similarities and train on them to produce better results. However, existing such techniques which focus on cross-media retrieval are currently limited due to binary constraints or optimization problems. These hashing techniques suffer from collected quantization errors and low quality long length hash codes. Some research efforts have managed to partially improve this problem but they suffer from other problems like being computationally expensive, not scalable to large-scale datasets and not applicable to cross-media search. The paper proposes a new framework for discrete hashing as a potential solution to these challenges.

The paper introduces a solution called Label Preserving Multimedia Hashing (LPMH). It uses semantic labels of the instances to optimize the binary hashes instead of pairwise distances. The paper demonstrates how this technique has a much better time complexity compared to others. It also demonstrates how its optimization is not tied to any specific form of loss function. It discusses how it can be applied to both single-media and cross-media applications. Finally, it proves how this technique is more scalable compared to others.

The technical contributions of the paper include providing a solution to the binary code inference problem and to the constrained optimization problem along with proofs. It provides work to demonstrate the combination of classification-based optimization with the new two stage learning framework for scalability. Finally, it runs experiments against multiple media search benchmarks.

II. TECHNIQUES

The technique introduced in this paper is called Label Preserving Multimedia Hashing (LPMH).

Label Preserving Multimedia Hashing (LPMH)

This technique uses a specific form of hashing to create a unique binary code for each piece of data and then uses these codes to sort and classify similar items. It is called label-preserving because these binary codes preserve the semantic label information that reflects the original labels of data, which is important in multimedia retrieval. It enables fast and accurate similarity search across media of different types including image, video, and text etc.

Stage 1: Binary Code Optimization

The goal of this stage is to preserve and learn the semantic labels in the binary code learning stage. A decision function is used to convert an input and output a decision while keeping dimensionality of the label vector the same. It achieves this by solving a Mixed Integer Programming (MIP) problem which alternately optimizes for weight matrix and binary codes. The objective is to minimize the loss functions. The outputted binary codes instead of hash functions can be either

discrete or continuous and make the optimization process more complex. Therefore it needs some modification and simplifications to make it manageable.

The optimization process is a complex task as it involves both discrete and continuous decision variables. It first focuses on fixing the binary codes. The paper makes this efficient by proposing subgradient descent methods or Stochastic Gradient Descent (SGD) as techniques for continuous optimization. Next it considers fixing the weight matrix by discrete optimization using a technique called Binary Integer Programming (BIP). It iteratively solves each bit by fixing all other bits and repeats for each bit. First it performs simplification to make the problem a general BIP problem and then applies a simple analytical solution.

The algorithm for this stage outlines the major steps and uses four loss functions Cross-Entropy Loss, Squared Loss, Logistic Loss and Hinge Loss. These loss functions are crucial since they measure the discrepancy between predicted and actual output and can be optimized to minimize the loss of labels from binary codes. However, the algorithm can work with any loss function.

1. Initially, it starts by random initialization of the binary codes (B) and weight matrix (W).
2. It enters an outer loop until convergence is reached or max number of iterations are hit. The inner loop updates the weight matrix (W). This loop estimates the gradient of the loss function with respect to W over a mini-batch of decision labels.
3. It then uses an update rule to update the weight matrix (W) in the gradient descent direction. It repeats the process for each iteration.
4. Following this, there is another inner loop that updates the binary codes (B) until convergence is reached. In each iteration, it computes a constant vector for each bit and solves for that bit-row of binary code according to a specific equation.
5. Finally it outputs the learned binary codes (B) and weight matrix (W).

The alternation between optimization of B and W gradually refines the parameters until it minimizes the loss function and still preserves the semantic label information. These binary codes can then be used to map new data samples from cross-media type to common Hamming codes. The weight matrix can be used for decision mapping.

Stage 2: Hash Function Learning

Bit balance constraints make the binary optimization a complex problem and is commonly ignored by hashing methods. However, including these constraints can improve the accuracy and results of the learned binary codes. Therefore, to solve the problem of complexity, the paper proposes an effective solution which involves flipping the bits to decrease the objective function's second term. This bit flipping can be done in one step due to its independence nature and O(n^2) time complexity. The proof for this polynomial complexity is outlined by the paper. Once hash codes are obtained, the framework needs to learn a set of hash functions for each media type to support cross-media search across a common Hamming space.

The following algorithm is used for balancing binary codes:

1. Initially, it starts by initialization of b as the sign of the constant vector. For each element in c, if it is positive, b will be -1 and if it is negative, b will be 1.
2. Next it computes the sign of unbalance by summing all elements of be and using the sum's sign.
3. Next, it computes the level of unbalance (m) by halving the sum of all elements of b. This is the degree to which b is unbalanced.
4. Next, it finds candidate bits (A) from b such that they all have the same sign as sign of unbalance (s) and its corresponding elements in C are less than lambda (λ). These bits can be flipped to make b more balanced.

5. Then, it compares the size of A with the level of unbalance (m) and finds m bits in A that correspond to the smallest elements in C. If the size of A > m, it flips the signs of these m bits, else flips the sign of all bits in A.

This algorithm can run on polynomial time and ensures that b has equal number of 1s and -1s to make it more balanced. Balancing makes the multimedia retrieval more efficient.

Each hash bit is considered a classification problem and can be used as a label to train a binary classifier. It uses Boosted Decision Trees (BST) to solve these classification problems due to their good nonlinear mapping, better generalization, efficient testing and fast scalability on large scale datasets.

Binary Classification is the first step by BST where each bit of the hash code is used as a binary label to train a classifier. A decision tree is trained to predict whether a bit should be 1 or -1 based on input features. This is followed by boosting. It trains a sequence of decision trees adaptively to adjust the incorrect classifications of the previous one. It does so by adjusting the weights of the samples based on previous predictions. The decision function is produced for each bit using an ensemble of trees and is related to a specific type of media. This allows new data of different media types to be mapped into a common Hamming space for enabling search. These decision functions are used while searching the tree to perform value comparison on the input and descend the tree depending on the output.

The linear complexity of algorithms in both stages and fast convergence of the proposed optimizations make the process of training highly efficient and scalable. In summary, this technique creates a system that can understand and categorize different media types while preserving their original labels. It learns on them and reduces their dimensionality of binary codes for simpler searching and produces media-specific hash functions to search these codes for all media types.

III. CRITICAL ANALYSIS

In the growing era of technology and social media, multimedia retrieval is a highly demanding and challenging notion. This paper introduces a technique that solves major challenges faced by the current techniques and the experiments produced optimistic results. This technique works for both single-media and cross-media applications which is essential in real world applications. It is efficient and scalable due to fast convergence algorithms. It preserves the semantic labels of the data in binary codes which is important in retrieving text based content. It employs the use of SGD and BIP optimization techniques which are effective and make the quality of results better.

The technique has certain limitations. It solves the MIP problem using effective solutions but this problem can still be computationally expensive. Moreover, it has a dependency on the quality of labels. If the labels are noisy or inaccurate, the learned binary codes may not reflect the original labels. Finally, the choice of loss functions can impact the performance of the algorithms and while a specific few have been mentioned, the optimal choice may depend on the requirements of the task at hand.

The proposed solution has room for more research. Future e-fforts can be aimed at exploring other optimization methods other than SGD and BIP that can improve the efficiency even further. There can be research to explore ways to handle noise or incomplete labels since they are quite common in real world data. There can also be efforts directed towards finding better ways to handle the bit balance constraint.

Overall, the authors have done a good job of finding and experimenting an effective framework for multimedia hashing. The paper also focuses over the most common challenges at every step and provides sufficient optimization and experimentation to address these.