Boosted Locality Sensitive Hashing: Discriminative, Efficient, and Scalable Binary Codes for Source Separation

Sunwoo Kim, *Student Member, IEEE*, and Minje Kim, *Senior Member, IEEE*

***Abstract*—We propose a novel adaptive boosting approach to learn discriminative binary hash codes, boosted locality sensitive hashing (BLSH), that can represent audio spectra efficiently. We aim to use the learned hash codes in the single-channel speech denoising task by designing a nearest neighborhood search method that operates in the hashed feature space. To achieve the optimal denoising results given the highly compact binary feature representation, our proposed BLSH algorithm learns simple logistic regressors as the weak learners in an incremental way (i.e., one by one) so that each weak learner is trained to complement the mistake its predecessors have made. Upon testing, their binary classification results transform each spectrum of noisy speech into a bit string, where the bits are ordered based on their significance, adding scalability to the denoising system. Simple bitwise operations calculate Hamming distance to find the K-nearest matching hashed frames in the dictionary of training noisy speech spectra, whose associated ideal binary masks are averaged to estimate the denoising mask for that test mixture. In contrast to the locality sensitive hashing method's random projections, our proposed supervised learning algorithm trains the projections such that the distance between the self-similarity matrix of the hash codes and that of the original spectra is minimized. Likewise, the process conceptually aligns to the Adaboost algorithm, although ours is specialized in learning binary features for source separation rather than classification. Experimental results on speech denoising suggest that the BLSH algorithm learns more discriminative representations than Fourier or mel spectra and the nonlinear kernels derived from them. Our compact binary representation is expected to facilitate model deployment onto resource-constrained environments, where comprehensive models (e.g., deep neural networks) are unaffordable.**

***Index Terms*—Speech Enhancement, Locality Sensitive Hashing, AdaBoost, Kernel Methods.**

I. Introduction

SOURCE separation is an essential module for many practical audio applications, e.g., in speech communication, automatic speech recognition, a hearing aid, etc. Deep learning-based source separation models have been proposed to improve the single-channel speech denoising and speech separation performance with performances nearing the ideal ratio masking results (IRM) [1]–[6], or sometimes exceeding them [7]–[10]. Despite the recent success of neural network architectures, deep learning applications come with huge memory size and high computational complexity which makes it intractable for deployment into resource-constrained devices, even just for feedforward inferences. For mobile and embedded applications that require speaker separation and noise cancellation in real life scenarios, it is crucial to solve this issue to ensure quality speech communication.

To solve this issue, there is ongoing research to reduce the model size and complexity while trying to preserve the accuracy of the models despite the reduction. Optimization of convolutional operators allowed for building small, low latency models. Namely, depthwise separable filters (MobileNet) [11], fire modules (SqueezeNet) [12], and group pointwise convolutions with channel shuffle operation (ShuffleNet) have been explored to create lightweight deep convolution neural networks [13]. Another popular approach to reducing model complexity is pruning less active weights [14], filters [15], [16], and even layers [17]. The other dimension of network compression is to reduce the number of bits to represent the network parameters, sometimes down to just one bit [18]–[21], one of its kind has shown promising performances in speech denoising [22], [23].

In this study we take another route to lightweight source separation by redefining the problem as a K-nearest neighborhood (KNN) search task: for a given test mixture, the separation is done by finding the nearest mixture spectra in the training set, and consequently their corresponding ideal binary mask (IBM) vectors [24]. However, the complexity of the search process linearly increases with the size of the training data and the frequency dimension of the spectrum. We expedite this tedious process by converting the query and database spectra into a hash code to exploit bitwise matching operations during the search process. To this end, we start from locality sensitive hashing (LSH), which is to construct hash functions such that similar data points are more probable to collide in the hashed space, or, in other words, more similar in terms of Hamming distance [25]–[28]. With increasing number of bits, the Hamming distance between binary hash codes will asymptotically approach the Euclidean distance between pairs of data. While simple and effective, the random projection-based nature of the LSH process

Manuscript received 10 February 2022; revised 21 June 2022; accepted 27 July 2022. Date of publication 5 August 2022; date of current version 19 August 2022. This work was supported by the National Science Foundation under Award Number 1909509. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Keisuke Kinoshita. *(Corresponding author: Minje Kim.)*

The authors are with the Department of Intelligent Systems Engineering, Indiana University, Bloomington, IN 47408 USA (e-mail: kimsunw@indiana.edu; minje@indiana.edu).

Digital Object Identifier 10.1109/TASLP.2022.3196877

is not trainable, thus limiting its performance when one uses it for a specific problem.

Under the requirement of having a small number of bits in the codebook, several authors have studied machine learning approaches to learn more compact codes such as semantic hashing [29] and spectral hashing [30], which are either with a computing-intensive conversion process (semantic hashing) or not scalable to high-dimensional or a large amount of data (spectral hashing). As an alternative, we propose a learnable, but still projection-based hash function, Boosted LSH (BLSH), so that the separation is done in the efficient binary space, which is learned in a data-driven way rather than relying on the random projections [31]–[33]. BLSH reduces the redundancy in the randomly generated LSH codes by relaxing the independence assumption among the projection vectors and learning them sequentially in a *boosting* manner such that they complement one another, an idea shown in search applications [32]–[35]. BLSH learns a set of linear classifiers (i.e., perceptrons), whose binary classification results serve as a hash code. To learn the sequence of binary classifiers we are based on the adaptive boosting (AdaBoost) technique [36], while we redefine the original classification-based AdaBoost algorithm so that it works on our separation problem. Since the binary representation is to improve the quality of the hash code-based $\boldsymbol{K}$NN search during the separation, the objective of our training algorithm is to maximize the representativeness of the hash codes by minimizing the loss between the two self-similarity matrices (SSM) constructed from the original spectra and from the hash codes.

BLSH can also be seen as an embedding technique with a constraint that the embedding has to be binary. Finding embeddings that preserve the semantic similarity is a popular goal in many disciplines. In natural language processing, Word2Vec [37] or GloVe [38] methods use pairwise metric learning to retrieve a distributed contextual representation that retains complex syntactic and semantic relationships within documents. Another model that trains on similarity information is the Siamese networks [39], [40] which learn to discriminate a pair of examples. Utilizing similarity information has also been explored in the source separation community by posing denoising as a segmentation problem in the time-frequency plane with an assumption that the affinities between time-frequency regions could condense complex auditory features together [41]. Inspired by studies of perceptual grouping [42], in [41] local affinity matrices were constructed out of cues specific to that of speech. Then, spectral clustering segments the weighted combination of similarity matrices to unmix speech mixtures. In our framework, we propose to maintain the original similarity among binary hash codes, so a test-time mixture frame can quickly find a few matching frames from the training dataset through the $\boldsymbol{K}$NN search. Then, from the matching training mixtures we can estimate the masking vector.

Our framework resembles the deep clustering method (DC) whose objective is to approximate the ideal pairwise affinity matrix induced from ideal binary masks [4]. DC utilizes a neural network encoder that produces discriminant spectrogram embeddings which are then partitioned to separate the speakers from a mixture. To make the partitioning feasible, e.g., by using k-means clustering, DC or its extension deep attractor network (DAN) learns an embedding space where each speaker's time-frequency units are concentrated around a centroid [3]. ChimeraNet extended the work by utilizing deep clustering as a regularizer for TF-mask approximation [43]. Our proposed framework also learns an embedding space, but instead of estimating an embedding vector per TF bin, we learn per-frame binary hash codes, which are then used to estimate masks. Moreover, given that the embedding vectors must be binary, we need to develop a learning algorithm that accommodates the discrete nature.

We evaluate BLSH on the single-channel denoising task and empirically show that with respect to the efficiency, our system compares favorably to deep learning architectures and generalizes well over unseen speakers and noises. Since binary codes can be cheaply stored and the $\boldsymbol{K}$NN search is expedited with bitwise operations, we believe this to be a good alternative for the speech enhancement task where efficiency matters.

This paper extends our preliminary study [44], where we proposed the initial BLSH version and $\boldsymbol{K}$NN search approach for speech denoising. In this journal version, we add new contributions summarized as follows:

- We provide an in-depth explanation of individual modules of the BLSH algorithm.
- We analyze the behavior of both LSH and BLSH frameworks on self-similarity targets generated from nonlinear feature extraction methods: mel scale and radial basis function transformations.
- We provide a thorough comparison against deep learning baselines including one of the state-of-the-art source separation models, Conv-TasNet [8].

The rest of the paper is organized as follows. We first describe the $\boldsymbol{K}$NN based source separation in Section II. Section III introduces the proposed BLSH algorithm, which aims to learn from various SSM targets, such as the ones constructed from short-time Fourier transform (STFT) and mel spectra as well as a nonlinear kernel function. Experimental setup is presented in Section IV. Evaluation results, discussion, and complexity analysis are presented in Section V. Final concluding remarks are given in Section VI.

II. The $\boldsymbol{K}$NN Search-Based Source Separation

In this section, we present the baseline $\boldsymbol{K}$NN search-based source separation frameworks for mask estimation, which finds the $\boldsymbol{K}$NNs in two different ways: using the spectral coefficients and LSH codes.

A. Baseline 1: Direct Spectral Matching

We assume that if two mixture spectra are similar, they consist of similar sources. Hence, their IBMs must be similar, too. The $\boldsymbol{K}$NN search-based source separation requires a large reference dictionary of training examples, whose IBMs are known ahead of time. For a given test example, the separation algorithm searches for only $\boldsymbol{K}$NN to the query spectra to infer its mask.

Algorithm 1: $\boldsymbol{K}$NN source separation.

1: Input: $\boldsymbol{x}$, $\boldsymbol{H}$, $\boldsymbol{Y}$
▷A test mixture vector, the dictionary, and IBMs
2: Output: $\hat{\boldsymbol{y}}$ ▷A denoising mask vector
3: Initialize an empty set $\mathcal{N} = \varnothing$ and $\mathcal{A}_{\text{min}} = 0$
4: **for** $t \leftarrow 1$ to T **do**
5: **if** $|\mathcal{N}| < \boldsymbol{K}$ **then**
6: Add t to $\mathcal{N}$
7: Update $\mathcal{A}_{\text{min}} \leftarrow \min_{k \in \mathcal{N}} \mathcal{S}_{\cos}(\boldsymbol{x}, \boldsymbol{H}_{:,k})$
8: **else if** $\mathcal{S}_{\cos}(\mathbf{x}, \mathbf{H}_{:,\mathbf{t}}) > \mathcal{A}_{\text{min}}$ **then**
9: Replace $\arg\min_{k \in \mathcal{N}} \mathcal{S}_{\cos}(\boldsymbol{x}, \boldsymbol{H}_{:,k})$ in $\mathcal{N}$ with t
10: Update $\mathcal{A}_{\text{min}} \leftarrow \min_{k \in \mathcal{N}} \mathcal{S}_{\cos}(\boldsymbol{x}, \boldsymbol{H}_{:,k})$
11: **end if**
12: **end for**
13: return $\hat{\boldsymbol{y}} \leftarrow \frac{1}{\boldsymbol{K}} \sum_{k \in \mathcal{N}} \boldsymbol{Y}_{:,k}$

Let $\boldsymbol{H} \in \mathbb{R}^{D \times T}$ be the normalized feature vectors from T frames of training mixture examples, e.g., noisy speech spectra. T can potentially be a very large number as it exponentially grows with the number of sources. Out of many potential choices, we are interested in STFT and mel spectra as the feature vectors. For example, if $\boldsymbol{H}$ is from magnitudes of STFT on the training mixture signals, D corresponds to the number of Fourier transform's subbands F, i.e., $D = F$, while for mel spectra $D < F$.

Columns of $\boldsymbol{H}$ are normalized with their L_2 norm. We also prepare their corresponding IBM vectors, $\boldsymbol{Y} \in \{0, 1\}^{F \times T}$, whose dimension F matches that of STFT, regardless whether we use STFT or mel spectra as input. For a test mixture spectrum, $\bar{\boldsymbol{x}} \in \mathbb{C}^F$, our goal is to estimate a denoising mask, $\hat{\boldsymbol{y}} \in \mathbb{R}^F$, to recover the source by masking, $\hat{\boldsymbol{y}} \odot \bar{\boldsymbol{x}}$. While masking is applied to the complex STFT spectrum $\bar{\boldsymbol{x}}$, the $\boldsymbol{K}$NN search can be done in the D-dimensional feature space $\boldsymbol{x} \in \mathbb{R}^D$, e.g., mel spectra or full Fourier coefficients.

Algorithm 1 describes the $\boldsymbol{K}$NN source separation procedure. We use notation $\mathcal{S}$ as the affinity function, which denotes the cosine similarity function with a subscript: $\mathcal{S}_{\cos}$. For a given pair of L_2-normalized D-dimensional feature vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, it is defined as an inner product:

$$\mathcal{S}_{\cos}(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a}^\top \boldsymbol{b} \tag{1}$$

For each frame $\boldsymbol{x}$ in the test-time mixture signal, we find the $\boldsymbol{K}$ closest frames in the reference database (line 4 to 9), which form the set of indices of $\boldsymbol{K}$NN, $\mathcal{N} = \{\tau_1, \tau_2, \ldots, \tau_{\boldsymbol{K}}\}$. Using them, we find the corresponding IBM vectors from $\boldsymbol{Y}$ and take their average (line 13).

In this proposed separation framework, finding $\boldsymbol{K}$ frames that *best* approximate the query ensures quality source separation. Although it is well known that simple Euclidean or cosine distance cannot represent the semantic similarity between high-dimensional data points, preserving the local similarity in the feature space and the SSM can lead to successful manifold learning [45]–[47]. In this regard, our separation method also requires the reference database to be sufficiently large to represent the manifold of the data distribution. However, a large T is burdensome not only for storage but also during the test time since the distance computation between $\boldsymbol{x}$ and $\boldsymbol{H}$ (floating-point inner product) is costly.

B. Baseline 2: LSH With Random Projections

We can reduce the storage overhead and expedite Algorithm 1 using hashed spectra and the Hamming similarity between them. For this second baseline, we follow the basic LSH setup that applies random projections to the data points followed by a step function (i.e., the sign function) as the hash function.

We define L random projection vectors as $\boldsymbol{P} \in \mathbb{R}^{L \times D}$ that are randomly initialized and fixed throughout the experiment. The l-th projection using $\boldsymbol{P}_{l,:}$ defines the l-th bit in the codes:

$$\tilde{\boldsymbol{H}}_{l,:} = \text{sgn}(\boldsymbol{P}_{l,:}\boldsymbol{H} + b_l) \tag{2}$$

where b_l is a bias term. Applying the same $\boldsymbol{P}$ onto $\boldsymbol{H}$ and $\boldsymbol{x}$, we obtain T training spectra's hash codes each of which is a bipolar binary bit string, i.e, $\tilde{\boldsymbol{H}} \in \{-1, +1\}^{L \times T}$, and one for the test spectrum, $\tilde{\boldsymbol{x}} \in \{-1, +1\}^L$, respectively. Accordingly, Hamming similarity also replaces the similarity function by counting the number of matching bits in the pair of binary feature vectors:

$$\mathcal{S}_{\text{Ham}}(\tilde{\boldsymbol{a}}, \tilde{\boldsymbol{b}}) = \sum_l \mathcal{I}(\tilde{a}_l, \tilde{b}_l)/L, \tag{3}$$

where $\mathcal{I}(\tilde{a}_l, \tilde{b}_l) = 1$ iff $\tilde{a}_l = \tilde{b}_l$, otherwise $\mathcal{I}(\tilde{a}_l, \tilde{b}_l) = 0$. Or, we can simplify this expression into an inner product by assuming bipolar binaries as follows:

$$\mathcal{S}_{\text{Ham}}(\tilde{\boldsymbol{a}}, \tilde{\boldsymbol{b}}) = \tilde{\boldsymbol{a}}^\top \tilde{\boldsymbol{b}}/L, \quad \text{if } \tilde{\boldsymbol{a}}, \tilde{\boldsymbol{b}} \in \{-1, +1\}^L \tag{4}$$

Hence, what we expect from the hash codes is that the Hamming similarity of an originally similar pair should be more likely to be high than that of a dissimilar pair. Suppose there is a similar pair in the original feature space, e.g., $\mathcal{S}_{\cos}(\boldsymbol{x}, \boldsymbol{H}_{:,i}) > \rho$, where ρ stands for a threshold. Then, a dissimilar pair can be also defined similarly: $\mathcal{S}_{\cos}(\boldsymbol{x}, \boldsymbol{H}_{:,j}) \leq \rho$. Their corresponding LSH codes are discriminative, if they fulfill the following inequality:

$$p\left(\mathcal{S}_{\text{Ham}}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{H}_{:,i}}) = 1\right) > p\left(\mathcal{S}_{\text{Ham}}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{H}_{:,j}}) = 1\right). \tag{5}$$

In other words, a successful LSH hashing step guarantees a higher chance of the originally similar pair colliding each other in the same bucket than a dissimilar pair.

Fig. 1 overviews the separation process of *Baseline 2*. First, the system prepares the hash codes of all mixture spectra dictionary $\tilde{\boldsymbol{H}}$. Then, it applies the same hashing process (i.e., the same projection matrix $\boldsymbol{P}$) to the test-time mixture spectrum to acquire its hash code $\tilde{\boldsymbol{x}}$. The $\boldsymbol{K}$NN search follows to find the similar training spectra, but the search is performed in the hash code space $\tilde{\boldsymbol{H}}$ instead of the original feature space $\boldsymbol{H}$. Using the found $\boldsymbol{K}$NN entries $\mathcal{N} = \{\tau_1, \tau_2, \ldots, \tau_{\boldsymbol{K}}\}$, the system infers the masking vector. Note that it is similar to *Baseline 1* described in 1, except that the search is done based on the simpler-to-compute Hamming similarity.

The downside of the $\boldsymbol{K}$NN search task performed in a binary feature space $\tilde{\boldsymbol{H}}$ is that it may result in a suboptimal search compared to $\boldsymbol{H}$. However, the upside is that the binary comparison

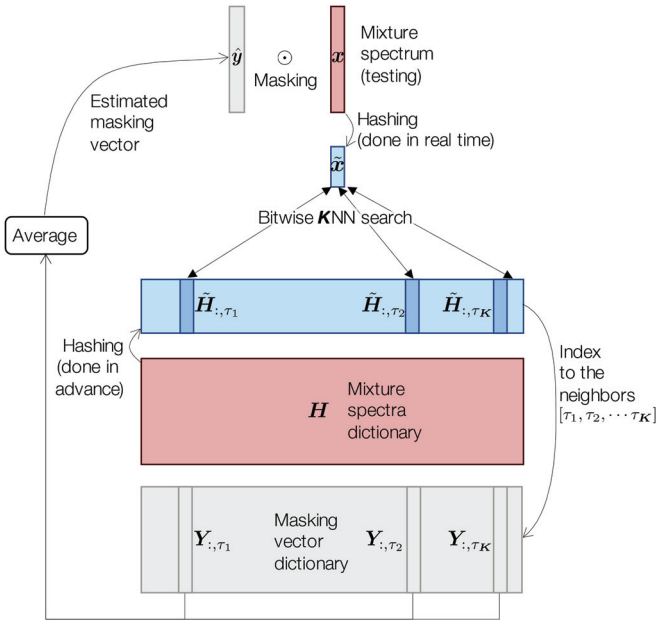


Fig. 1. The $\boldsymbol{K}$NN-based source separation process using hash codes.

can be fast and efficient since it can be performed using efficient bitwise operations (e.g., bitwise AND and pop counting) with supporting hardware. Furthermore, if the size of the hash table L is much smaller than the original input features, more items can be loaded into memory, resulting in disk I/O cost reduction.

Since the $\boldsymbol{K}$NN search will be on the hash codes, the search performance depends on the quality of the hash function as to how well it preserves the original similarity after hashing. However, when it comes to the LSH method, the lackluster quality of the codes originates from the data-blind nature of random projections [29], [30]. Instead of attempting to learn the best projection vectors from data, the representativeness of the hash codes relies greatly on the randomness in the projection vectors: during the initial construction, each hash function, i.e., the projection, is chosen independently and uniformly at random. Hence, a large L is required to form discriminative hash codes that can guarantee high precision. This is detrimental in terms of query time, computational cost of projecting the query to hash codes, and storage overhead from the large number of projections, even though they result in binary codes.

III. Boosted LSH Training Algorithm

The proposed BLSH algorithm addresses the inefficiency of the LSH algorithm by *learning* each projection vector. With the boosting concept, we learn the projections in an incremental fashion, i.e., one by one in the order of their significance. In this way, each projection improves the total representativeness of the code by trying to fix the preceding projections' mistakes in approximating the original similarity function. As a result, we can expect a hash code whose bits are ordered based on their contribution to the search performance, adding scalability to the entire system. For example, if a too large L is not affordable, instead of re-initializing the entire projection table $\boldsymbol{P}$, one can trim the end of the existing table. To summarize, the goals of the BLSH training algorithm are as follows:

- *Compactness:* By using the boosting concept, we aim at achieving higher representation power using shorter bit strings in comparison with ordinary LSH codes.
- *Scalability:* Out of the total L bits, the system can choose to utilize only the first few bits because the bits are ordered based on their significance. Hence, the system can adapt to different computational environments with greater scalability.
- *Efficiency:* The inference on the proposed hashing process should still be affordable (i.e., a single-layer neural network) for efficient test-time hashing.

In this section, we present our learnable LSH algorithm for source separation. We extend LSH-based $\boldsymbol{K}$NN search with approximating SSMs, boosting, and kernel functions.

A. Binary Approximations of Similarity Matrices

Rather than employing a large number of independent random projections, we aim to *learn* similarity-preserving hash functions that better fit the data distribution. Our goal of learning to hash is to preserve the features' discriminative properties as shown in the literature [29], [30], [48], [49]. Among them, deep learning models [29], [49] have been proposed to learn data-dependent hash codes as well; nonetheless, the computational complexity of multilayered neural networks is restrictive in hashing applications that operate under a limited resource budget.

The proposed BLSH algorithm is still based on the projection and sign function as in the LSH algorithm. However, instead of relying on random projections, we see each projection as a weak classifier and learn its model parameters, i.e., the projection vector, during training. Here, we will use the same formulation defined in (2), while the projection vector $\boldsymbol{P}_{l,:}$ and bias b_l are trainable parameters. Note that the non-differentiable sign function results in an intractable Dirac delta function during the gradient descent computation. We circumvent this issue by employing the derivative of the hyperbolic tangent function as a surrogate of the delta function:

$$\operatorname{sgn}'(x) \approx \tanh'(x) = 1 - \tanh^2(x). \tag{6}$$

During training the projection vectors are directed to minimize the discrepancies between the pairwise affinity relationships among the original features in $\boldsymbol{H}$ and those we construct from their corresponding hash codes $\tilde{\boldsymbol{H}}$. We express the loss function in terms of the dissimilarity between the target self-similarity matrices (SSM) and its binary estimation. We denote $\boldsymbol{S} \in \mathbb{R}^{T\times T}$ for the target SSM,

$$\boldsymbol{S}_{i,j} = \mathcal{S}_{\cos}(\boldsymbol{H}_{:,i}, \boldsymbol{H}_{:,j}) \tag{7}$$

where $i, j \leq T$. Suppose that the optimization process is learning the l-th projection $\boldsymbol{P}_{l,:}$, which result in $\tilde{\boldsymbol{H}}_{l,:}$ bipolar binary hash codes. We construct the binary SSM $\tilde{\boldsymbol{S}}$ as

$$\tilde{\boldsymbol{S}}_{i,j} = \frac{\tilde{\boldsymbol{H}}_{l,i} \cdot \tilde{\boldsymbol{H}}_{l,j} + 1}{2} \tag{8}$$

Algorithm 2: BLSH training.

Input: $\boldsymbol{S} \in \mathbb{R}^{T\times T}$ ▷Target SSM
Output: $\boldsymbol{P} \in \mathbb{R}^{L\times D}$ ▷Set of projections
$\boldsymbol{W} \in \mathbb{R}^{T\times T} \leftarrow$ uniform vector of $\frac{1}{T\times T}$
$\boldsymbol{P} \in \mathbb{R}^{L\times D} \leftarrow$ random numbers
$\boldsymbol{\beta} \in \mathbb{R}^{L} \leftarrow 0$
for $l \leftarrow 1$ to L **do**
  $\boldsymbol{P}_{l,:} \leftarrow \arg\min_{\boldsymbol{P}_{l,:}} \sum_{i,j} \mathcal{D}(\tilde{\boldsymbol{S}}_{i,j} \parallel \boldsymbol{S}_{i,j}) \odot \boldsymbol{W}_{i,j}^{(l)}$
  $\varepsilon_l = \sum_{i,j} \mathcal{D}(\tilde{\boldsymbol{S}}_{i,j} \parallel \boldsymbol{S}_{i,j}) \odot \boldsymbol{W}_{i,j}^{(l)}$
  $\beta_l \leftarrow \ln((1-\varepsilon_l)/\varepsilon_l)$
  $\boldsymbol{W}_{i,j}^{(l+1)} = \boldsymbol{W}_{i,j}^{(l)} \exp(\beta_l \sum_{i,j} \mathcal{D}(\tilde{\boldsymbol{S}}_{i,j} \parallel \boldsymbol{S}_{i,j}))$
end for
return $\boldsymbol{P}$

where we shift and scale such that $\tilde{\boldsymbol{S}} \in \{0,1\}^{T\times T}$. Then, our objective is to minimize the dissimilarities:

$$\sum_{i,j} \mathcal{D}\left(\tilde{\boldsymbol{S}}_{i,j} \parallel \boldsymbol{S}_{i,j}\right) \tag{9}$$

The objective also depends on the choice of the distance metric $\mathcal{D}$, which is cross-entropy in our case. With this objective the learned binary hash codes can be more compact and representative than the ones obtained from random projections as in (2). There can be potentially many different solutions to this optimization problem, such as solving this optimization directly for the set of projection vectors $\boldsymbol{P}$ or spectral hashing that learns the hash codes directly with no assumed projection process [30]. We present our proposed method in the following subsection.

B. Boosted LSH

A major drawback of LSH is the independence between the random hyperplanes. Hence, LSH tends to result in redundancy in number of projections, i.e., a large L value, to build discriminative hash codes. As we saw in Section III-A, we resolve this issue by turning LSH into a learnable hashing process, which minimizes the gap between SSM and BSSM (9). In this way, the hash codes behave similarly to the already-known discriminative features $\boldsymbol{H}$. A remaining issue though is that the BSSM approximation has to pre-define the number of projections L, within which there is no straightforward order of significance. Moreover, if L is set to be too large, some projections do not contribute to the discrimination, leading to a cost in test-time inference, while they increase the bit depth of the code.

We address this issue by proposing a *boosted* LSH algorithm, where each projection is learned one by one, in the order of their significance. Hence, one can rely more on the initially learned projections than the later added ones. This property is designed to handle the test-time use case, where the application chooses to use only a small number of most significant projections, i.e., the *first few* bits from the code to save the processing time and resources.

To this end, we reformulate AdaBoost, one of the most widely studied adaptive boosting strategy which was originally proposed for classification [36]. It learns efficient weak learners, e.g., a linear classifier, in an additive manner: each new weak learner complements those learned in previous rounds and infers something new about the data. The set of weak learners constructs an adaptive basis function model, whose weighted sum makes the strong final prediction. In such a weighted sum, the complementary nature tends to give larger weights to the earlier learners than the later added ones.

The AdaBoost framework aligns with our applicational goal, because we also need such an order of importance between the projections. Moreover, since each of the hash code bits is a result of a weak classifier, i.e., linear combination followed by a sign function, the hashing process is similar to the weighted sum of weak learners in AdaBoost. The biggest difference between AdaBoost and BLSH is that BLSH does not use misclassification as the loss. Instead, it aims to learn weak learners whose binary decision results can improve the speech enhancement performance.

Consequently, we propose to adjust the loss function and the boosting algorithm accordingly. Here, since our speech enhancement is based on the matching process between test and training examples, we once again seek the binary feature space where matching results are similar to the ones in the original feature space. Hence, our training objective is to approximate the pairwise relationship between the known discriminative features $\boldsymbol{H}$ using the binary features $\tilde{\boldsymbol{H}}$. It leads to the total error function defined as follows:

$$\sum_{i,j} \mathcal{D}\Big(\sum_{l=1}^{L} \beta_l \tilde{\boldsymbol{S}}_{i,j} \parallel \boldsymbol{S}_{i,j} \Big), \tag{10}$$

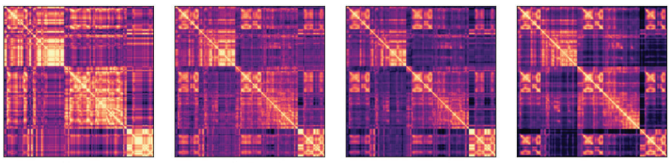
where β_l denotes the weight of the l-th weak learner to the total estimation and the l-th weak learner is defined by the l-th projection $\tilde{\boldsymbol{H}}_{:,i} = \boldsymbol{P}_{l,:}\boldsymbol{H}_{:,i} + b_l$ that constructs the shifted and scaled BSSM $\tilde{\boldsymbol{S}}_{i,j}$ (8).

In AdaBoost, when each weak learner is trained, it focuses more on the previously misclassified examples. This mechanism is implemented by a weighting scheme on the samples: those misclassified training samples receive an exponentially large weight, so that the next weak learner pays more attention to them. Likewise, the per-sample weights in AdaBoost are defined over all training samples.

On the other hand, in BLSH for speech denoising, the optimization process involves T^2 pairwise relationships even though it begins with T training examples. It is because our goal is to preserve the cross-sample correlation rather than a per-sample property. Hence, we redesign AdaBoost's sample weighting scheme by defining a *per-pair* symmetric weight matrix $\boldsymbol{W} \in \mathbb{R}^{T\times T}$. It is initialized uniformly, but then updated after adding every projection. The weight matrix is designed to exponentially increase its values when a pair (i,j) is incorrectly represented by BSSM compared to the target SSM as follows:

$$\boldsymbol{W}_{i,j}^{(l+1)} = \boldsymbol{W}_{i,j}^{(l)} \exp\left(\beta_l \sum_{i,j} \mathcal{D}\left(\tilde{\boldsymbol{S}}_{i,j} \parallel \boldsymbol{S}_{i,j}\right)\right), \tag{11}$$

where the superscript (l) denotes the per-pair weight matrix associated with the l-th weak learner.



(a) (b) (c) (d)

Fig. 2. Self-affinity matrices of varying L hash codes and original time-frequency bins. (a) $L = 10$ (b) $L = 50$ (c) $L = 300$ (d) Ground-truth SSM.

As in AdaBoost, the effect of this update rule is to exaggerate the importance of BSSM cells that failed to reconstruct the target SSM. The $(l+1)$-th weak learner can then use these weights to concentrate on *hard-to-approximate* pairs of examples. This approach pursues a complementary $(l+1)$-th projection, thus overall rapidly increasing the approximation quality in (10) with relatively smaller L projections. The final boosted objective for the l-th projection is formulated as

$$\varepsilon_l = \sum_{i,j} \mathcal{D}\left(\tilde{\boldsymbol{S}}_{i,j} \parallel \boldsymbol{S}_{i,j}\right) \odot \boldsymbol{W}_{i,j}^{(l)}. \tag{12}$$

Under this boosted objective, the first *few* weak learners result in binary features that approximate the majority of the original feature similarity, thereby dramatically reducing the overall storage of projections, computation from projecting elements, and the length of hashed bit strings. Given the learned l-th projection and per-pair weights, we obtain the projection weights as

$$\beta_l = \ln \frac{1-\varepsilon_l}{\varepsilon_l}. \tag{13}$$

Although the cross-entropy loss $\mathcal{D}$ is relatively small, we clip $\varepsilon_l = \min(\varepsilon_l, 1)$ prior to computing (13).

Algorithm 2 summarizes the BLSH procedure introduced in this section. Fig. 2 shows the complementary nature of the projections and their convergence behavior. After learning the projection matrix $\boldsymbol{P}$, the rest of the test-time source separation process is the same as described in Section II-B.

C. Kernel Functions as Similarity Measure

Deep neural networks can undoubtedly produce highly discriminative data-dependent hash codes via a series of nonlinear transformations. For example, in semantic hashing, autoencoders are trained to learn codes in its bottleneck layer [29]. However, the deep architecture tends to require heavy inference computation as opposed to BLSH. Meanwhile, it is also true that the projection-based LSH process corresponds only to a shallow neural network limiting its performance.

In this section we employ a nonlinear kernel to produce the target SSM so that the proposed BLSH algorithm learns from a potentially very complex pairwise relationship. Based on the learning objective defined in (9), the relative performance of the learned hash functions significantly depends on the definition of ground-truth targets (e.g., distance metric) [30]. Hence, the quality of the feature vectors used to construct the target SSM is critical. For example, if $\boldsymbol{H}$ stands for the raw spectrogram of STFT magnitudes, we could derive a more abstract feature via a nonlinear feature transform function $\phi(\cdot)$. Although the transform function can be a tractable one, such as a neural network encoder, in this section we assume that it can be intractable. By using the kernel trick, we implement our new objective by computing the pairwise similarity through the kernel function:

$$\mathcal{G}\left(\boldsymbol{H}_{:,i}, \boldsymbol{H}_{:,j}\right) = \boldsymbol{\phi}(\boldsymbol{H}_{:,i})^\top \boldsymbol{\phi}(\boldsymbol{H}_{:,j}). \tag{14}$$

For example, the previous formulation (9) can be considered as the case with an identity mapping function $\boldsymbol{\phi} = \boldsymbol{I}$ that forms a linear kernel. Among various kernels, we adopt the radial basis function (RBF) to create pairwise similarity embedded in high dimensional spaces [50]. An RBF kernel is defined on our normalized input features as follows:

$$\mathcal{G}\left(\boldsymbol{H}_{:,i}, \boldsymbol{H}_{:,j}\right) = C \exp\left(\frac{\boldsymbol{H}_{:,i}^\top \boldsymbol{H}_{:,j}}{\sigma^2}\right) \tag{15}$$

where $C = \exp(-1/\sigma^2)$ and $\mathcal{G}(\boldsymbol{H}_{:,i}, \boldsymbol{H}_{:,j}) \in [0,1]$ for normalized features. The hyperparameter σ^2 controls the width of the Gaussian-shaped decision boundary, which determines the locality of the similarity measure defined in the feature space [51]. Since RBF kernels can be represented as an implicit sum over an infinite sequence of polynomials, it nonlinearly defines an infinitely high-dimensional feature space ($\mathbb{R}^n \rightarrow \mathbb{R}^\infty$). Note that with the tractable RBF kernel function in (15) one can avoid the intractability involved in the infinitely high-dimensional feature transformation.

Using the kernel trick, we train another version of our hash functions. This time, the objective is to minimize the dissimilarity between the predicted BSSM and the RBF kernel defined between pairwise training feature vectors $\boldsymbol{H}$. With a shortened notation for the kernel matrix $\boldsymbol{G}_{i,j} = \mathcal{G}(\boldsymbol{H}_{:,i}, \boldsymbol{H}_{:,j})$ the loss function simply replaces the target SSM $\boldsymbol{S}$ of (10)-(12) with $\boldsymbol{G}$.

Note that the nonlinearity of the target SSM $\boldsymbol{G}$ can be adjusted using the hyperparameter σ^2: the smaller the value, the more local the distance in the feature space. Eventually, if $\boldsymbol{G}$ is strictly defined with local pairs, $\boldsymbol{G}$ introduces more nonlinearity. Likewise, σ^2 allows for flexibility in determining the target complexity of the system. We will investigate the impact of different choices of σ^2 in the experiment section.

IV. Experimental Setup

A. Datasets

To investigate the effectiveness of the proposed framework for speech enhancement, we evaluate our model on the single-channel speech denoising setup. A training set consisting of 10 hours of noisy utterances and their corresponding clean speech signals. The clean utterances are from the randomly selected but gender-balanced 160 speakers from the train fold of the TIMIT corpus. They are mixed with different non-stationary noise signals with 0 dB signal-to-noise ratio (SNR), namely {birds, casino, cicadas, computer keyboard, eating chips, frogs, jungle, machine guns, motorcycles, ocean} [52]. Since there are 10 short utterances per speaker, it amounts to 1,600 utterances recorded with a 16 kHz sampling rate. The projections are

learned on the training set and the output hash codes are saved as the dictionary.

One hour of evaluation data set was mixed from 110 unseen speakers in the TIMIT test set and unseen noise sources from the DEMAND database, which consists of domestic, office, indoor public places, transportation, nature, and street (**open mixture set**) [53]. The test set mixtures are with -5, 0, 5, and 10 dB SNR and gender-balanced as well.

We further test on the VoiceBank-DEMAND dataset by Valentini et al. [54] to evaluate our system on mixtures from unseen speakers and sentences.

We apply a short-time Fourier transform (STFT) with a Hann window of 1024 samples and a hop size of 256. Optionally, mel spectrograms are created using 128 mel scale filters. During training, the whole mixture speech was segmented into a mini-batch with a length of 1000 frames, such that the self-similarity matrices and the kernel matrices are of a reasonable size.

For evaluation, we calculate the signal-to-distortion ratio (SDR) [55]. We also report results in terms of model size, scale-invariant SDR (SI-SDR) [56], perceptual evaluation of speech quality (PESQ) with values ranging from -0.5 to 4.5 [57], and extended short-time intelligibility (ESTOI) [58].

B. Models

In our experiments we use a few different setups to provide a comprehensive understanding of the proposed model. First, we differentiate the models based on the type of the input features $\boldsymbol{x}$ as follows:

- $\texttt{IN}_{\texttt{STFT}}$: The 513-dimensional magnitude coefficients of STFT.
- $\texttt{IN}_{\texttt{mel}}$: The 128-dimensional mel spectra.

In addition, we also denote the models based on what they are targeting:

- SSM: This is the case of using the regular SSM but its behavior varies depending on the input representation, i.e., STFT or mel.
- $\texttt{RBF}_{\sigma^2}$: The case when RBF kernels are used as target. To compute RBF kernels, we only use STFT as input to discern the effects of nonlinearity introduced by the RBF kernel. The RBF kernels vary depending on the choice of σ^2.

We differentiate our hashing-based models based on two important hyperparameters. First, the number of projections L defines the bit depth of the binary hash codes. Our goal is to achieve a better performance using the proposed BLSH algorithm than the random projection-based LSH method if L is the same. Second, the size of the training dictionary matters. Although in theory we can use the entire training examples and convert them into the hash codes, using these T examples can increase the complexity of the $\boldsymbol{K}$NN search. Instead, we investigate a subsampling option, where we use only a proportion of T training examples, i.e., $\rho = 0.01$ or 0.1. Consequently, we also denote the two hashing mechanisms distinctively:

- $\texttt{LSH}_{L,\rho}$: The ordinary LSH-based method defined with L projections and ρT training examples.
- $\texttt{BLSH}_{L,\rho}$: Ditto, except that the hash codes are learned via the proposed BLSH algorithm.
- $\texttt{KNN}_{\rho}$: This one is a $\boldsymbol{K}$NN-based system, but using raw input features, STFT or mel, instead of the hash codes. Hence, it is not elaborated with the choice of L.

In all three $\boldsymbol{K}$NN methods, we fixed the neighborhood size to $\boldsymbol{K} = 10$.

For example, $\texttt{IN}_{\texttt{STFT}}\texttt{-RBF}_{\sigma^2=0.1}\texttt{-BLSH}_{L=300,\rho=0.1}$ denotes a $\boldsymbol{K}$ NN-based model that uses hash codes learned from the proposed BLSH method. It uses only 10% of the training data for the $\boldsymbol{K}$NN search and the bit depth is 300. When BLSH algorithm learns the projection it targets a RBF kernel with $\sigma^2 = 0.1$. The algorithm operates on the magnitudes of STFT spectra. Or, $\texttt{IN}_{\texttt{mel}}\texttt{-KNN}_{\rho=0.01}$ is a model that performs $\boldsymbol{K}$ NN search directly on the mel spectra. For cases applying to all L or both ρ values, we remove the subscripts from BLSH for brevity.

Finally, three neural network models are trained for comparison.

- FC: We train fully-connected (FC) network with two or three hidden layers. The number of hidden units ranges from 32 to 1,024.
- BiLSTM: A recurrent neural network using bidirectional long short-term memory (BiLSTM) [59] cells are employed. We use two BiLSTM layers by varying the number of memory cells and hidden units from 32 to 1,024.

Both models are trained with STFT magnitude spectrogram inputs and IBM targets. The logistic activation function was applied on the outputs of both models. For the FC network, batch normalization [60] and ReLU activation functions [61] were applied to the intermediate outputs. The learning rate was set as 1×10^{-4} for both FC and BiLSTM models. Adam optimizer was used for both models [62].

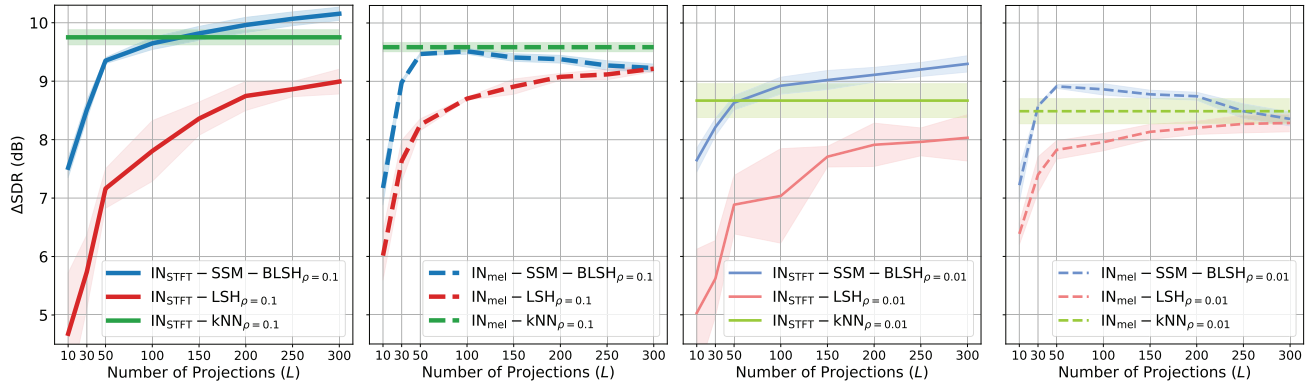
In addition, we also present another end-to-end model architecture that performs speech enhancement in the time domain as a reference. Conv-TasNet is a fully convolutional model that performs the masking operation in the latent space, achieving state-of-the-art performance in speech separation benchmarks [8]. While there have been a few other models that followed up showing meaningful improvement on speech separation [9], [10], [63], [64], we adopted the Conv-TasNet model as a stable reference. To train Conv-TasNet, we employed a much larger dataset consisting of Librispeech's training fold [65] and the MUSAN dataset's Free Sound corpus [66] by mixing them with a range of mixing ratio from -5 to 10 dB. Note that this training set is deliberately chosen to maximize the performance of the Conv-TasNet architecture, while the other models mentioned so far have been trained on a much smaller dataset.

V. Evaluation Results and Discussion

In this section, we compare the effects of different $\boldsymbol{K}$NN systems, relative subset size ρ, bit depth L, and the type of input features and training targets. Experimental results are reported using the evaluation data set (open mixture set).

A. Analysis of the Denoising Systems on STFT Features

From Fig. 3(a), we can observe that the proposed BLSH system clearly outperforms the LSH version in terms of average SDR improvements for both mel and STFT input features and



(a) $\texttt{IN}_{\texttt{STFT}}$ systems at $\rho = 0.1$ (b) $\texttt{IN}_{\texttt{mel}}$ systems at $\rho = 0.1$ (c) $\texttt{IN}_{\texttt{STFT}}$ systems at $\rho = 0.01$ (d) $\texttt{IN}_{\texttt{mel}}$ systems at $\rho = 0.01$

Fig. 3. Average and standard deviations (shades) of ΔSDR for various $\boldsymbol{K}$NN systems with respect to the number of projections L under different input features and ρ values. The KNN baseline is expressed as a solid horizontal line since it is not dependent on L. BLSH systems were trained on SSM targets.

different choices of ρ and L. With a sufficiently large L, $\texttt{IN}_{\texttt{STFT}}$-SSM-BLSH$_{L\approx100,\rho=0.1}$ can reach the $\texttt{IN}_{\texttt{STFT}}$-KNN$_{\rho=0.1}$'s performance. For all L values, the performance of $\texttt{IN}_{\texttt{STFT}}$-SSM-BLSH$_{\rho=0.1}$ is consistently higher than the random LSH version, $\texttt{IN}_{\texttt{STFT}}$-LSH$_{\rho=0.1}$. This demonstrates that the boosting mechanism provide more discriminative features compared to its counterpart using random hash codes.

It is also promising that the BLSH hash codes surpass the performance of the raw Fourier coefficients, while LSH hash codes fall short. It should be noted that the $\texttt{IN}_{\texttt{STFT}}$-KNN$_{\rho=0.1}$ baseline conducts the $\boldsymbol{K}$ NN search on high-dimensional floating-point spectrum coefficients (i.e., $D = F = 513$), whereas the BLSH systems are based on the bitwise Hamming distance between L-bit hash codes.

B. Analysis of the Denoising Systems on Mel Features

We can observe from Fig. 3(b) that $\texttt{IN}_{\texttt{mel}}$-LSH achieves better performance compared to $\texttt{IN}_{\texttt{STFT}}$-LSH, demonstrating the effectiveness of mel features to construct discriminative hash codes even when a limited bit depth L is allowed (i.e., less than $L = 150$). The mel features' improved performance is expected because mel scaling performs a nonlinear feature transform, a logarithmic scaling process that aligns better to human auditory perception. This nonlinear feature transform adds additional representativeness to the LSH methods that rely solely on randomly initialized projections.

However, the mel transformation is a lossy dimension reduction method that condenses multiple subbands into one. Although perceptually motivated, mel scaling eliminates vital information. This fact is reflected in Fig. 3(b). Our proposed boosting-based framework, $\texttt{IN}_{\texttt{mel}}$-SSM-BLSH$_{\rho=0.1}$, performs best when using only up to a handful of hash codes (e.g., $L = 50$). The scores saturate after $L = 50$ with no further improvements and rather exhibit overfitting behavior. Hence, we believe that the proposed BLSH algorithm learns the majority of features using approximately the first $L = 50$ weak learners. This result contrasts with the BLSH results on STFT features in Fig. 3(a), where more hyperplanes learned via the BLSH algorithm keep improving the denoising performance.

Hence, the experimental results suggest that the BLSH method on the raw STFT features is more useful than the LSH counterparts: given the same bit depth, BLSH hash codes provide better discrimination, and consequently, better separation. Mel features give boosts to the LSH models, but BLSH on STFT still outperforms it significantly.

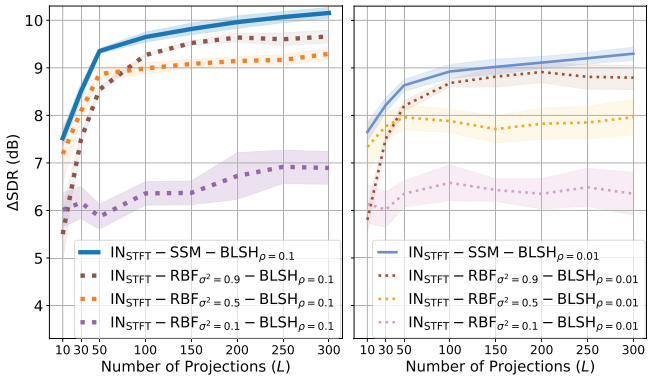
C. The Impact of Subsampling the Training Set

In Fig. 3(c) and (d), we observe a global performance drop of all three $\boldsymbol{K}$NN systems. First, the baseline $\texttt{IN}_{\texttt{STFT}}$-kNN$_{\rho=0.01}$ exhibits more than 1 dB loss by reducing the training set from 10% to 1% of the original size. While the BLSH systems also show decreased performance, they are more robust to the subsampling process. Random projections of LSH show a noticeable sensitivity to the randomness in the subsampling procedure: the standard deviation is significantly larger especially in the $\rho = 0.01$ case, whereas that of BLSH remains tighter. Also, now the BLSH systems start to exceed the baseline when $L = 50$. Note that when $L = 50$, $\texttt{IN}_{\texttt{mel}}$-SSM-BLSH$_{L=50,\rho=0.01}$ shows slightly better performance than $\texttt{IN}_{\texttt{STFT}}$-SSM-BLSH$_{L=50,\rho=0.01}$, showcasing mel spectrum's usefulness in this extreme condition. Overall, a similar trend to the $\rho = 0.1$ cases is retained: we still prefer the BLSH model trained from STFT features the best.

D. The Impact of Using RBF Kernels as the Training Targets

In addition to the SSMs computed by the mel or STFT features, we also evaluate the validity of nonlinear kernels as a learning target of our BLSH algorithm. Considering the nonlinearity introduced during mel scaling, we construct RBF kernels from the raw STFT features to adequately measure the impact of the level of nonlinearity controlled by the kernel width parameter, σ^2.

Fig. 4(a) compares the SSM-based BLSH results $\texttt{IN}_{\texttt{STFT}}$-SSM-BLSH$_{\rho=0.1}$ with three BLSH systems targeting RBF kernels with different kernel widths, $\sigma^2 = 0.1$, 0.5, and 0.9. We observe that RBF-based systems exhibit saturation in performance and are unable to catch up to $\texttt{IN}_{\texttt{STFT}}$-SSM-BLSH$_{\rho=0.1}$. Specifically, wider kernel widths, $\sigma^2 = 0.9$ and 0.5, provide



(a) $\rho = 0.1$ (b) $\rho = 0.01$

Fig. 4. Average and standard deviations of ΔSDR for BLSH systems trained on SSM and RBF targets under varying σ^2 and ρ values. Only the STFT feature was used for each of the systems.

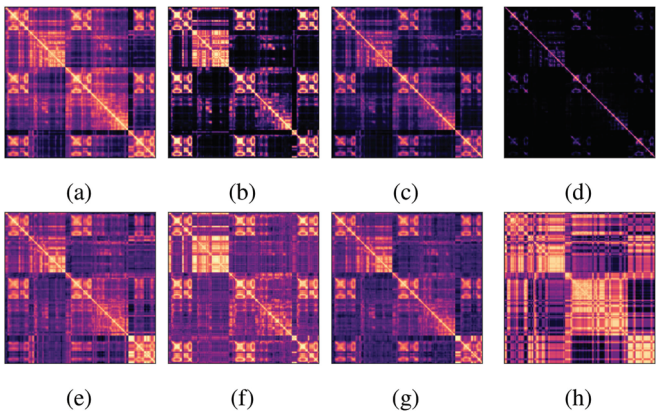


Fig. 5. Self-affinity matrices and their estimations from binary codes for varying transformations. The binary outputs shown are constructed as the weighted sum of BSSMs created from all $L = 300$ projections. (a) STFT SSM (b) Mel SSM (c) RBF $\sigma^2 = 0.5$ SSM (d) RBF $\sigma^2 = 0.1$ SSM (e) STFT BSSM (f) Mel BSSM (g) RBF $\sigma^2 = 0.5$ BSSM (h) RBF $\sigma^2 = 0.1$ BSSM.

reasonable performance. However, they still fall short compared to $\texttt{IN}_{\texttt{STFT}}\texttt{-SSM-BLSH}_{\rho=0.1}$, indicating that the RBF kernels used here are either too linear to introduce additional distinction between codes or the nonlinear relation is far from perceptual similarity between audio spectra. Furthermore, the most extreme case, $\texttt{IN}_{\texttt{STFT}}\texttt{-RBF}_{\sigma^2=0.1}\texttt{-BLSH}$, rarely shows improvement even during the earlier phases of training. This is due to the narrow width of the kernel which only emphasizes elements that are relatively close and harshly devalues the scores of distant frames to near-zero values. Smaller σ^2 allows us to focus on local features, thus being more nonlinear. However, it removes too much information and restricts the perceptrons from learning any more features. Similar trends are observed when the subsampling rate is reduced to $\rho = 0.01$ in Fig. 4(b), with even less relevant results from the mid-sized kernel ($\sigma^2 = 0.5$).

Fig. 5 gives a more detailed view to the varying relationship between BSSM and SSM depending on how the target SSM is constructed. From BSSM reconstructions, first we can see that the linear kernel computed from STFT coefficients is easy for the BLSH algorithm to reconstruct (Fig. 5(a) vs. (e)), which is also the best denoising solution we have achieved so far. Meanwhile, the mel SSM target shown in Fig. 5(b) exhibits more contrast than Fig. 5(a), indicating more locality introduced in the similarity between mel spectra. BSSM mimics this relationship in Fig. 5(f), while it fails to catch up with the drastically dissimilar pairs: dark pixels in Fig. 5(b) are not well represented in the BSSM reconstruction.

The RBF kernels also introduce similar locality to the target SSMs. While with $\sigma^2 = 0.5$ the RBF kernel behaves as an SSM with a certain level of nonlinearity (Fig. 5(c)), the most narrow kernel width $\sigma^2 = 0.1$ suppresses most of the non-local pairwise similarities. As a result, $\texttt{IN}_{\texttt{STFT}}\texttt{-RBF}_{\sigma^2=0.1}\texttt{-BLSH}$ (Fig. 5(h)) is distinctly different from its target (Fig. 5(d)), demonstrating the difficulty for binary codes to approximate the sparsity in the target SSM. This is reflected by the respective denoising performances (e.g., $\texttt{IN}_{\texttt{STFT}}\texttt{-RBF}_{\sigma^2=0.1}\texttt{-BLSH}$ shows relatively low improvements for increasing L). We believe that it is because the RBF kernels do not properly represent the perceptual relationship between audio spectra.

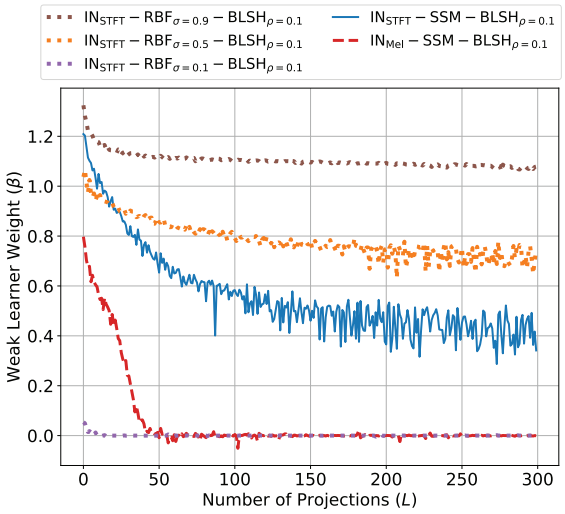


Fig. 6. Weak learner weights (β) for BLSH systems trained on varying features and kernels.

E. The Learning Behavior of the BLSH Algorithm

In our BLSH algorithm, similarly to the ordinary boosting method, the β values represent relative importance of each weak learner (i.e., the projection) in the final ensemble. It is common to see them diminish as more weak learners are added because the initial weak learners are more important than the later learned ones.

Fig. 6 indeed shows the decreasing β values in various BLSH learning processes. First, we note here that the β values learned from the mel spectra $\texttt{IN}_{\texttt{mel}}\texttt{-SSM-BLSH}_{\rho=0.1}$ decays very fast in the initial few rounds and then stagnates, while the STFT graph from $\texttt{IN}_{\texttt{STFT}}\texttt{-SSM-BLSH}_{\rho=0.1}$ shows more steady decrease. Hence, this comparison explains the saturating or overfitting behavior of the BLSH algorithm on the mel spectrum input shown in Fig. 3(b) and (d).

TABLE I
MODEL SIZE, AVERAGE SDR, SI-SDR, PESQ AND ESTOI FOR VARIOUS SYSTEMS EVALUATED ON OPEN MIXTURE SET. HASHING-BASED $\boldsymbol{K}$NN SYSTEMS ARE REPORTED USING OPTIMAL L PARAMETERS, AND THEIR MODEL SIZE IS EXPRESSED AS A SUM OF REFERENCE DATABASE SIZE AND MODEL SIZE. FC AND BiLSTM MODELS ARE REPORTED USING RELEVANT PARAMETERS N_l AND N_h, THE NUMBER OF LAYERS AND HIDDEN UNITS RESPECTIVELY, IN PLACE OF D AND L. SIMILARLY CONV-TASNET MODEL IS REPORTED USING N_r AND N_b DENOTING THE NUMBER OF REPEATS AND BLOCKS RESPECTIVELY. WE USED THE SAME PARAMETERS AS REPORTED IN THE CONV-TASNET PAPER. THE MODEL SIZES ARE REPORTED USING SINGLE-PRECISION (I.E. 32 BITS PER PARAMETER) FOR BOTH $\boldsymbol{K}$NN AND DEEP LEARNING SYSTEMS

System	Feature	ρ	L	Model Size (MB)	SDR (dB)	SI-SNR (dB)	PESQ	ESTOI(%)
Mixture	-	-	-	-	0.07	0.00	1.51	81.97
IBM	-	-	-	-	20.38	19.98	2.93	94.58
KNN	STFT	0.1	-	503.55	9.77	8.76	1.80	73.74
	Mel	0.1	-	125.64	9.61	8.12	1.73	74.60
	STFT	0.01	-	50.36	8.65	7.58	1.76	71.38
	Mel	0.01	-	16.79	8.36	6.85	1.66	71.20
LSH	STFT	0.1	300	9.20 + 0.62	8.97	7.96	1.79	72.50
	Mel	0.1	300	1.53 + 0.15	9.23	7.72	1.73	73.60
	STFT	0.01	300	0.92 + 0.62	7.95	6.90	1.76	71.24
	Mel	0.01	300	0.15 + 0.15	8.25	6.71	1.68	71.20
BLSH	STFT	0.1	300	9.20 + 0.62	10.18	9.10	1.81	74.98
	Mel	0.1	50	1.53 + 0.03	9.47	7.84	1.84	76.58
	STFT	0.01	300	0.92 + 0.62	9.31	8.15	1.73	72.56
	Mel	0.01	50	0.15 + 0.03	8.92	7.25	1.80	75.78
		N_l	N_h					
FC	STFT	3	1024	12.61	13.10	12.34	1.64	82.32
			256	1.58	12.92	12.08	1.74	82.10
			64	0.30	12.19	11.28	1.55	80.40
			32	0.14	11.74	10.51	1.59	79.64
		2	1024	8.41	10.29	9.85	1.68	78.46
			256	1.32	10.25	9.71	1.59	77.06
			64	0.28	10.01	9.35	1.53	74.53
			32	0.14	9.89	9.07	1.48	74.18
BiLSTM		2	1024	155.36	14.74	14.24	1.87	86.74
			256	13.68	14.18	13.62	1.86	85.79
			64	1.85	13.50	12.91	1.83	84.69
			32	0.79	13.06	12.21	1.81	82.89
		N_r	N_b					
Conv-TasNet	Raw	3	8	19.94	18.48	18.19	2.45	92.47

From Fig. 6, it is also noticeable that targeting too localized RBF kernels is detrimental. When $\sigma^2 = 0.1$, for example, the β values quickly drop and become insignificant. SSMs created using RBF kernels with small kernel width σ^2 are mostly comprised of near-zero values (e.g., Fig. 5(d)), and most of the information can be learned using only a few weak learners. For these SSM targets, adding on more weak learners do not contribute to the final boosted solution, which is depicted by $\beta \to 0$ values for increasing L. The diminishing β values show that the newly added weak learners no longer learn significant information and the hyperplanes are not discriminative.

F. Comparison Among $\boldsymbol{K}$NN Systems

Table I shows the model size and performance of all three $\boldsymbol{K}$NN systems: KNN, LSH, and BLSH. We narrow our discussion to BLSH systems trained on linear SSM targets. Model sizes are computed as the total memory occupied by the training dictionary and hash functions. The KNN baselines show the enormous memory occupied by the dictionary. For example, at $\rho = 0.1$ subsampling rate, $\texttt{IN}_{\texttt{STFT}}\texttt{-KNN}_{\rho=0.1}$ requires $\sim$0.5 GB. $\texttt{IN}_{\texttt{mel}}\texttt{-KNN}_{\rho=0.1}$ requires less since the number of subbands for mel spectra is lower than STFT's. LSH methods show significant reduction in dictionary size with minimal storage overhead from the random projections, although at the cost of drop in performance. Across various subsampling rates ρ, LSH methods show a distinct drop in performance, especially in terms of SDR and SI-SDR, compared to its KNN counterparts using the same input features.

Our proposed BLSH method consistently outperforms the LSH approaches and also KNN on the STFT magnitude features. $\texttt{IN}_{\texttt{STFT}}\texttt{-SSM-BLSH}_{L=300,\rho=0.1}$ achieves better scores than $\texttt{IN}_{\texttt{STFT}}\texttt{-KNN}_{\rho=0.1}$ across all metrics. As discussed previously in Section V-A, this demonstrates the better representativeness of the learned hash codes than the raw Fourier coefficients or the basic LSH codes. Furthermore, our BLSH methods using mel spectra features can save more memory than LSH by requiring smaller L. $\texttt{IN}_{\texttt{mel}}\texttt{-SSM-BLSH}_{L=50}$ achieves higher scores with fewer hash functions than $\texttt{IN}_{\texttt{mel}}\texttt{-LSH}_{L=300}$ for both $\rho = 0.1$ and 0.01 cases.

Next, we evaluate the computational complexity of the aforementioned $\boldsymbol{K}$NN systems and highlight the efficiency of BLSH models. First, the baseline KNN search procedure requires a linear scan of all real valued feature vectors in $\boldsymbol{H}$, giving $O(QDT)$, where Q stands for the floating-point precision (e.g., $Q = 64$ for double-precision). This procedure is restrictive since T needs to be large for quality source separation and the distance computation (i.e. floating-point inner product) is costly, which is the reason for pursuing an expedited approach to using hash codes.

Since the LSH baseline also uses Algorithm 1, the dependency to T remains the same. However, we can reduce the complexity from $O(QDT)$ to $O(LT)$ if $L < QD$. The procedure can be significantly accelerated with supporting hardware as the Hamming similarity calculation is done through bitwise operations. Also, the size of the hash table is designed to be much smaller than the original input features ($L < QD$) and hence can be loaded into memory, resulting the disk I/O cost reduction.

Our proposed BLSH framework does not change the runtime complexity $O(LT)$ of baseline 2. Nevertheless, BLSH can outperform LSH with smaller L thanks to the boosting mechanism (e.g., $\texttt{IN}_{\texttt{mel}}\texttt{-SSM-BLSH}_{L=50,\rho=0.1}$ scores higher than any LSH systems using $L = 300$). In conclusion, BLSH models can achieve higher accuracy and more efficient memory usage than both baseline systems, computationally heavy KNN and data-blind LSH approaches.

G. Comparison Against Deep Neural Networks (DNN)

Although our goal in this work is not to compete with the state-of-the-art DNN's performance, as our solution is more about maximizing the utility of the binary models, here we present some denoising results from various DNN models to provide a reference point.

In Table I, we also compare the BLSH system with deep neural networks, a fully-connected (FC) network and a bi-directional long short-term memory (BiLSTM) network using varying number of hidden units. We denote N_l and N_h as the number of hidden layers and units respectively. We also report the performance of one of the state-of-the-art source separation models, Conv-TasNet. We use N_r and N_b to denote the number of *repeats* and *blocks*, which defines the internal separator module's architecture in Conv-TasNet; we used the default values as reported in the paper.

First, it is obvious that neural networks with complex structure (e.g., BiLSTM and Conv-TasNet) easily outperform BLSH models by all evaluation metrics. We observe that Conv-TasNet can achieve better performance than the largest BiLSTM while using significantly smaller architecture (19.94 vs 155.36 MB). Given that these models are either designed to learn from a long sequence (i.e., BiLSTM) or capable of processing a relatively long segment of input signal (i.e., 1-second input for Conv-TasNet), comparison to BLSH's frame-by-frame processing results is inadequate. Meanwhile, it also signifies that the proposed BLSH methods do not scale up to provide state-of-the-art performance. Our aim in designing the BLSH method was to maximize its performance in the hashing-based denoising paradigm.

Given the drastic difference in model sizes, a more reasonable comparison would be between BLSH and the small neural network models. For example, BLSH outperforms FC models of various architectures and performs similarly to BiLSTM models at least in terms of PESQ. Furthermore, BLSH can achieve comparable ESTOI scores to small FC models. For example, the ESTOI score of $\texttt{IN}_{\texttt{mel}}\texttt{-SSM-BLSH}_{L=50,\rho=0.01}$ with 0.18 MB of parameters is better than the 2×64 FC model with 0.28 MB (75.78% vs 74.18%).

From the perspective of a neural network, the BLSH or LSH methods can be seen as a shallow neural network with only one hidden layer, whose activations are binarized. Hence, it is expected that BLSH does not compete with the properly trained DNNs, especially in terms of SDR or SI-SDR. Still, we believe that BLSH may work as a reasonable solution. For example, in comparison to the similar-sized FC networks, BLSH loses only about 7.6% ($\texttt{IN}_{\texttt{mel}}\texttt{-SSM-BLSH}_{L=50,\rho=0.1}$'s 9.47 dB SDR improvement vs. FC 2×256's 10.25 dB) or 9.8% ($\texttt{IN}_{\texttt{mel}}\texttt{-SSM-BLSH}_{L=50,\rho=0.01}$'s 8.92 dB SDR improvement vs. FC 2×32's 9.89 dB) of the denoising performance. Nonetheless, we note that model compression techniques can be applied to the baseline DNN models. Prior works [23], [67], [68] demonstrated effective model compression methods can still achieve decent performance with a high compression ratio. Applying such compression pipelines to our baseline DNN models would potentially reduce the model sizes down to similar levels as our BLSH systems while still maintaining their superior denoising performances.

Considering that these DNN models are operating with floating-point values, a hardware support that benefits from bitwise operations can favor BLSH, as it clearly outperforms the other $\boldsymbol{K}$NN-based baselines, KNN and LSH. In our current experimental setup with $T = 2453960$ frames from the TIMIT dataset, a $\texttt{IN}_{\texttt{mel}}\texttt{-SSM-BLSH}_{L=50,\rho=0.01}$ would require $50 \times 24539 \approx 1.2$ M bitwise operations to process a single frame. For a 1-second input, or 63 frames, the system would require $\sim$77 M total bitwise operations. In comparison, a small 2×64 RNN model takes $\sim$13.7 M MACs (multiply-and-accumulate operations) for a 1-second input. Bitwise operations provide speedups over floating-point multiplication and accumulation, and it has been shown to achieve more than $\sim$50x speedup [69] when inference is run on CPU. Considering the expedited inference speed, our $\texttt{IN}_{\texttt{mel}}\texttt{-SSM-BLSH}_{L=50,\rho=0.01}$ system would potentially run faster than the small RNN model. However, note that the runtime analysis of a binary operation-only system highly depends on the hardware setup.

Furthermore, BLSH systems use significantly less computation during training compared to a small DNN model. Each projection is a vector of length F, so the total number of trainable parameters is $F \times L$. For a $\texttt{IN}_{\texttt{mel}}\texttt{-SSM-BLSH}_{L=50,\rho=0.01}$ system, this adds up to $128 \times 50 = 6.4$ K parameters, which is significantly less than that of a small 2×32 FC model which consists of 34.6 K trainable parameters. In addition, BLSH provides ordered binary representations based on their contribution, which adds scalability to the system as another merit.

H. Evaluation on Different Test Conditions

Table II shows the performance of BLSH systems on the same open mixture set scaled at SNR values -5, 5 and 10 dB, which are different from the noise signals with 0 dB SNR in the training set. Our proposed BLSH method shows consistent enhancement results in terms of SDR, SI-SDR, and PESQ under the different SNR conditions. As with the 0 dB noisy mixture set shown in Table I, we again see a decrease in ESTOI from the enhanced results. A noticeable difference is that for less noisy environments (5 and 10 dB), $\texttt{IN}_{\texttt{mel}}\texttt{-SSM-BLSH}$ does not show better ESTOI than $\texttt{IN}_{\texttt{STFT}}\texttt{-SSM-BLSH}$ as it did under harsher conditions (-5 and 0 dB).

Table III shows the results from testing our BLSH solutions on unseen speakers and sentences from the Voicebank-DEMAND dataset [54]. Our proposed system trained on TIMIT speakers does not perform well on the unseen speaker condition. $\texttt{IN}_{\texttt{STFT}}\texttt{-SSM-BLSH}_{L=300,\rho=0.1}$ shows a slight improvement of

TABLE II
SDR, SI-SDR, PESQ AND ESTOI FOR BLSH SYSTEMS EVALUATED ON OPEN MIXTURE SET SCALED AT -5, 5, AND 10 DB SNR VALUES, DIFFERENT FROM THE 0 DB SNR USED TO GENERATE THE TRAINING MIXTURE SET

System	Feature	ρ	L	SDR (dB)	SI-SNR (dB)	PESQ	ESTOI(%)
Mixture	-	-	-	-4.85	-5.00	1.29	72.12
BLSH	STFT	0.1	300	6.30	5.20	1.47	62.03
	Mel	0.1	50	6.42	4.97	1.53	66.86
	STFT	0.01	300	5.70	4.48	1.51	61.14
	Mel	0.01	50	5.84	4.40	1.48	65.69
Mixture	-	-	-	5.05	5.00	1.81	88.87
BLSH	STFT	0.1	300	13.23	12.16	2.16	83.70
	Mel	0.1	50	11.66	9.84	2.14	82.57
	STFT	0.01	300	12.38	11.15	2.16	82.03
	Mel	0.01	50	11.27	9.37	2.10	82.18
Mixture	-	-	-	10.04	10.00	2.22	93.61
BLSH	STFT	0.1	300	15.50	14.40	2.56	88.47
	Mel	0.1	50	13.21	11.25	2.49	86.04
	STFT	0.01	300	14.57	13.25	2.56	87.23
	Mel	0.01	50	12.84	10.73	2.48	85.87

TABLE III
SDR, SI-SDR, PESQ AND ESTOI FOR BLSH SYSTEMS EVALUATED ON THE VOICEBANK-DEMAND DATASET [54]

System	Feature	ρ	L	SDR (dB)	SI-SNR (dB)	PESQ	ESTOI(%)
Mixture	-	-	-	9.32	8.58	1.99	87.91
BLSH	STFT	0.1	300	8.67	8.83	1.81	76.43
	Mel	0.1	50	7.36	7.61	1.89	75.96
	STFT	0.01	300	8.06	8.24	1.86	75.88
	Mel	0.01	50	6.75	6.99	1.89	75.30

0.25 dB Δ SI-SDR, but it does not show improvements on other metrics. Other systems are unable to enhance the noisy mixtures but rather decrease their quality.

Likewise, the proposed BLSH systems are with a limited generalization power. It can generalize well to unseen noise levels quite successfully, while on test mixtures of unseen speakers and noise sources, its performance is limited. The weakness comes from the small training datasets that are not representative enough. Also, the compact model size prohibits the system from learning robust features for general-purpose speech enhancement.

VI. CONCLUSION

In this paper, we proposed a data-driven hashing algorithm for the source separation problem using nearest neighbor search. Under this setup, the $\boldsymbol{K}$-nearest matching frames in the dictionary to the test frame are found, and a denoising mask estimated by the average of associated IBMs. To implement a lightweight solution, we utilize locality sensitive hash functions to transform the query and database spectra into binary hash codes. In doing so, memory space is reduced and search process expedited using bitwise matching operations. We compress the framework further through a learnable approach to hashing using the boosting theory. The hash functions are learned sequentially as linear classifiers in a boosting manner such that they complement one another. Our learning objective is set to minimize the discrepancy between the self-similarity of real valued spectra and hash codes. Hence, we preserve the original similarity between the hash codes, so a test query can be quickly matched from the database through the bitwise $\boldsymbol{K}$NN search. Our framework trains binary classifiers sequentially under a boosting paradigm, culminating to a better approximation of the original self-similarity matrix with shorter hash strings. With learned projections and expedited inference using bitwise operations, our method offers a lightweight solution to the speech enhancement problem.

To study the level of nonlinearity the learned hash codes achieves, we employed RBF kernels as the self-similarity targets and taught the weak learners to learn more complex pairwise relationships. Since the features are expected to be more discriminant through the nonlinear transformation implied by the RBF kernel, it is expected that the BLSH code learned from it may encode a certain level of nonlinear similarity. While the BLSH algorithm does achieve reasonable performance by targeting the RBF kernels, we also found that the data-blind nature of RBF kernel itself has only limited capacity in capturing perceptual relationship among audio spectra.

Evaluation results demonstrate that the $\boldsymbol{K}$NN performance with the learned hash codes is better than with random codes from locality sensitive hashing and even raw Fourier coefficients or mel scaled spectra. This shows that the hash codes learned through the proposed BLSH method function as more discriminative features than the real valued features. Our proposed framework is not comparable to comprehensive deep learning models in terms of denoising quality, but it showed promising results compared against a very small neural network with similar model size to the BLSH dictionary. In the future we will investigate an extension to temporally meaningful versions so that the model can handle a sequence of audio frames.[1]

[1]The open-sourced code and sound examples can be found at https://saige.sice.indiana.edu/research-projects/BWSS-BLSH.

REFERENCES

[1] A. Narayanan and D. L. Wang, "Ideal ratio mask estimation using deep neural networks for robust speech recognition," in *Proc. 2013 IEEE Int. Conf. Acoust. Speech and Signal Processing.*, 2013, pp. 7092–7096.

[2] D. L. Wang and J. Chen, "Supervised speech separation based on deep learning: An overview," *IEEE/ACM Trans. Audio, Speech Lang. Process.*, vol. 26, no. 10, pp. 1702–1726, Oct. 2018.

[3] Z. Chen, Y. Luo, and N. Mesgarani, "Deep attractor network for single-microphone speaker separation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2017, pp. 246–250.

[4] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2016, pp. 31–35.

[5] D. Yu, M. Kolbæk, Z.-H. Tan, and J. Jensen, "Permutation invariant training of deep models for speaker-independent multi-talker speech separation," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process.*, 2017, pp. 241–245.

[6] M. Kolbæk, D. Yu, Z.-H. Tan, and J. Jensen, "Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 25, no. 10, pp. 1901–1913, Oct. 2017.

[7] K. Zhen, M. S. Lee, and M. Kim, "Efficient context aggregation for end-to-end speech enhancement using a densely connected convolutional and recurrent network," 2019, *arXiv:1908.06468*.

[8] Y. Luo and N. Mesgarani, "Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 8, pp. 1256–1266, Aug. 2019.

[9] Y. Luo, Z. Chen, and T. Yoshioka, "Dual-path RNN: Efficient long sequence modeling for time-domain single-channel speech separation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2020, pp. 46–50.

[10] C. Subakan, M. Ravanelli, S. Cornell, M. Bronzi, and J. Zhong, "Attention is all you need in speech separation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2021, pp. 21–25.

[11] A. G. Howard et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

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

[13] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6848–6856.

[14] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *Proc. Int. Conf. Learn. Representations*, 2016.

[15] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. Int. Conf. Learn. Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=rJqFGTslg

[16] J. Lin, Y. Rao, J. Lu, and J. Zhou, "Runtime neural pruning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 2181–2191.

[17] Z. Wu et al., "Blockdrop: Dynamic inference paths in residual networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8817–8826.

[18] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 525–542.

[19] D. Soudry, I. Hubara, and R. Meir, "Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 963–971.

[20] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients," 2016, *arXiv:1606.06160*.

[21] M. Kim and P. Smaragdis, "Bitwise neural networks," in *Proc. Int. Conf. Mach. Learn. Workshop Resource-Efficient Mach. Learn.*, 2015.

[22] M. Kim and P. Smaragdis, "Bitwise neural networks for efficient single-channel source separation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 701–705.

[23] S. Kim, M. Maity, and M. Kim, "Incremental binarization on recurrent neural networks for single-channel source separation," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process.*, 2019, pp. 376–380.

[24] D. Wang, "On ideal binary mask as the computational goal of auditory scene analysis," in *Speech Separation by Humans and Machines*. Boston, MA, USA: Springer, 2005, pp. 181–197.

[25] P. Indyk and R. Motwani, "Approximate nearest neighbor – towards removing the curse of dimensionality," in *Proc. Annu. ACM Symp. Theory Comput.*, 1998, pp. 604–613.

[26] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. 20th Annu. Symp. Comput. Geometry*, 2004, pp. 253–262.

[27] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proc. 34th Annu. ACM Symp. Theory Comput.*, 2002, pp. 380–388.

[28] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *Proc. 47th Annu. IEEE Symp. Found. Comput. Sci.*, 2006, pp. 459–468.

[29] R. Salakhutdinov and G. Hinton, "Semantic hashing," *Int. J. Approx. Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[30] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1753–1760.

[31] Z. Li, H. Ning, L. Cao, T. Zhang, Y. Gong, and T. S. Huang, "Learning to search efficiently in high dimensions," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 1710–1718.

[32] G. Shakhnarovich, "Learning task-specific similarity," Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, MA, USA, 2005.

[33] G. Shakhnarovich, P. Viola, and T. Darrell, "Fast pose estimation with parameter-sensitive hashing," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2003, vol. 3 pp. 750–750.

[34] X. Liu, C. Deng, Y. Mu, and Z. Li, "Boosting complementary hash tables for fast nearest neighbor search," in *Proc. AAAI Conf. Artif. Intell.*, 2017, vol. 31, no. 1.

[35] H. Xu, J. Wang, Z. Li, G. Zeng, S. Li, and N. Yu, "Complementary hashing for approximate nearest neighbor search," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 1631–1638.

[36] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. Int. Conf. Mach. Learn.*, vol. 96. Citeseer, 1996, pp. 148–156.

[37] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*.

[38] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proc. Empirical Methods Natural Lang. Process.*, 2014, pp. 1532–1543. [Online]. Available:http://www.aclweb.org/anthology/D14-1162

[39] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Proc. ICML Deep Learn. Workshop*, 2015, vol. 2.

[40] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a 'siamese' time delay neural network," in *Prioc. Adv. Neural Inf. Process. Syst.*, 1994, pp. 737–744.

[41] F. R. Bach and M. I. Jordan, "Learning spectral clustering, with application to speech separation," *J. Mach. Learn. Res.*, vol. 7, pp. 1963–2001, 2006.

[42] M. Cooke and D. P. Ellis, "The auditory organization of speech and other sources in listeners and computational models," *Speech Commun.*, vol. 35, no. 3–4, pp. 141–177, 2001.

[43] Y. Luo, Z. Chen, J. R. Hershey, J. Le Roux, and N. Mesgarani, "Deep clustering and conventional networks for music separation: Stronger together," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2017, pp. 61–65.

[44] S. Kim, H. Yang, and M. Kim, "Boosted locality sensitive hashing: Discriminative binary codes for source separation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2020, pp. 106–110.

[45] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, vol. 14, pp. 585–591.

[46] M. Balasubramanian, E. L. Schwartz, J. B. Tenenbaum, V. de Silva, and J. C. Langford, "The Isomap algorithm and topological stability," *Science*, vol. 295, p. 7, Jan. 2002.

[47] S. T. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2326, 2000.

[48] D. Zhang, J. Wang, D. Cai, and J. Lu, "Self-taught hashing for fast similarity search," in *Proc. 33rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2010, pp. 18–25.

[49] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2064–2072.

[50] J.-P. Vert, K. Tsuda, and B. Schölkopf, "A primer on kernel methods," *Kernel Methods Comput. Biol.*, vol. 47, pp. 35–70, 2004.

[51] Q. Chang, Q. Chen, and X. Wang, "Scaling Gaussian RBF kernel width to improve SVM classification," in *Proc. Int. Conf. Neural Netw. Brain*, 2005, vol. 1, pp. 19–22.

[52] Z. Duan, G. J. Mysore, and P. Smaragdis, "Online PLCA for real-time semi-supervised source separation," in *Proc. Int. Conf. Latent Variable Anal. Signal Separation*, 2012, pp. 34–41.

[53] J. Thiemann, N. Ito, and E. Vincent, "The diverse environments multi-channel acoustic noise database (demand): A database of multichannel environmental noise recordings," in *Proc. Meetings Acoust.*, 2013, vol. 19, Art. no. 035081.
[54] C. Valentini-Botinhao, X. Wang, S. Takaki, and J. Yamagishi, "Investigating RNN-based speech enhancement methods for noise-robust text-to-speech," in *Proc. SSW*, 2016, pp. 146–152.
[55] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, no. 4, pp. 1462–1469, Jul. 2006.
[56] J. Le Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "Sdr–half-baked or well done?" in *Proc. ICASSP IEEE Int. Conf. Acoust., Speech Signal Process.*, 2019, pp. 626–630.
[57] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, "Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2001, vol. 2, pp. 749–752.
[58] J. Jensen and C. H. Taal, "An algorithm for predicting the intelligibility of speech masked by modulated noise maskers," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 24, no. 11, pp. 2009–2022, Nov. 2016.
[59] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.
[60] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[61] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1026–1034.
[62] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.
[63] N. Zeghidour and D. Grangier, "Wavesplit: End-to-end speech separation by speaker clustering," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 29, pp. 2840–2849, 2021.
[64] J. Chen, Q. Mao, and D. Liu, "Dual-path transformer network: Direct context-aware modeling for end-to-end monaural speech separation," in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, 2020.
[65] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2015, pp. 5206–5210.
[66] D. Snyder, G. Chen, and D. Povey, "MUSAN: A music, speech, and noise corpus," 2015, *arXiv:1510.08484*.
[67] Y. Luo, C. Han, and N. Mesgarani, "Ultra-lightweight speech separation via group communication," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2021, pp. 16–20.
[68] K. Tan and D. Wang, "Towards model compression for deep learning based speech enhancement," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 29, pp. 1785–1794, 2021.
[69] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Enabling ai at the edge with xnor-networks," *Commun. ACM*, vol. 63, no. 12, pp. 83–90, 2020.

Sunwoo Kim (Student Member, IEEE) received the B.S. degree in engineering physics from the University of Illinois at Urbana-Champaign,Champaign, IL, USA, in 2016. Since 2016, he has been working toward the Ph.D. degree in intelligent systems engineering with Indiana University, Bloomington, IN, USA. He works with Professor Minje Kim, Signals and Artificial Intelligence Group in Engineering. His research interests include efficient and scalable machine learning models for audio applications.

Minje Kim (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign, in 2016. He is currently an Assistant Professor of Intelligent Systems Engineering, Indiana University, Bloomington, IN, USA. He is also an Amazon Visiting Academic. Before that, he was a Researcher with Electronics and Telecommunications Research Institute, Daejeon, South Korea, from 2006 to 2011. His research interests include machine learning and audio signal processing. He is a Member of the IEEE AASP TC. He was the recipient of the NSF CAREER Award (2021), IEEE SPS Best Paper Award (2020), Google and Starkey's grants for outstanding student papers at ICASSP 2013 and 2014, respectively.