A. Proof of Lemma 1

For convenience, we re-state Lemma 1 here and then give the proof.

Lemma. *For any vector $x \in \mathbb{R}^d$, and any matrix $A \in \mathbb{R}^{m\times d}$ ($m < d$) with rank m, there exists an $\tilde{A} \in \mathbb{R}^{m\times d}$ with all singular values being ones, such that the following two ℓ_1-norm minimization problems have the same solution:*

$$P_1: \quad \min_{x'\in\mathbb{R}^d} \|x'\|_1 \quad \text{s.t. } Ax' = Ax. \tag{16}$$

$$P_2: \quad \min_{x'\in\mathbb{R}^d} \|x'\|_1 \quad \text{s.t. } \tilde{A}x' = \tilde{A}x. \tag{17}$$

Furthermore, the projected subgradient update of P_2 is given as

$$x^{(t+1)} = x^{(t)} - \alpha_t(I - \tilde{A}^T\tilde{A})\text{sign}(x^{(t)}), \quad x^{(1)} = \tilde{A}^T\tilde{A}x.$$

A natural choice for $\tilde{A}$ is $U(AA^T)^{-1/2}A$, where $U \in \mathbb{R}^{m\times m}$ can be any unitary matrix.

Proof. To prove that P_1 and P_2 give the same solution, it suffices to show that their constraint sets are equal, i.e.,

$$\{x : Ax = Az\} = \{x : \tilde{A}x = \tilde{A}z\}. \tag{18}$$

Since $\{x : Ax = Az\} = \{z + v : v \in \text{null}(A)\}$ and $\{x : \tilde{A}x = \tilde{A}z\} = \{z + v : v \in \text{null}(\tilde{A})\}$, it then suffices to show that A and $\tilde{A}$ have the same nullspace:

$$\text{null}(A) = \text{null}(\tilde{A}). \tag{19}$$

If v satisfies $Av = 0$, then $U(AA^T)^{-1/2}Av = 0$, which implies $\tilde{A}v = 0$. Conversely, we suppose that $\tilde{A}v = 0$. Since U is unitary, $AA^T \in \mathbb{R}^{m\times m}$ is full-rank, $(AA^T)^{(1/2)}U^T\tilde{A}v = 0$, which implies that $Av = 0$. Therefore, (19) holds.

The projected subgradient of P_2 has the following update

$$x^{(t+1)} = x^{(t)} - \alpha_t(I - \tilde{A}^T(\tilde{A}\tilde{A}^T)^{-1}\tilde{A})\text{sign}(x^{(t)}), \tag{20}$$

$$x^{(1)} = \tilde{A}^T(\tilde{A}\tilde{A}^T)^{-1}\tilde{A}z \tag{21}$$

Since $\tilde{A} = U(AA^T)^{-1/2}A$, we have

$$\begin{aligned}\tilde{A}\tilde{A}^T &= U(AA^T)^{-1/2}AA^T(AA^T)^{-1/2}U^T \\ &= U(AA^T)^{-1/2}(AA^T)^{1/2}(AA^T)^{1/2}(AA^T)^{-1/2}U^T \\ &= I. \end{aligned} \tag{22}$$

Substituting (22) into (21) gives the desired recursion:

$$x^{(t+1)} = x^{(t)} - \alpha_t(I - \tilde{A}^T\tilde{A})\text{sign}(x^{(t)}), \quad x^{(1)} = \tilde{A}^T\tilde{A}z.$$

□

B. Training parameters

Table 4 lists the parameters used to train ℓ_1-AE in our experiments. We explain the parameters as follows.

- Depth: The number of blocks in the decoder, indicated by T in Figure 1.
- Batch size: The number of training samples in a batch.
- Learning rate: The learning rate for SGD.
- $N_{\max}$: Maximum number of training epochs.
- $N_{\text{validation}}$: Validation error is computed every $N_{\text{validation}}$ epochs. This is used for early-stopping.
- $N_{\text{no improve}}$: Training is stopped if the validation error does not improve for $N_{\text{no improve}} * N_{\text{validation}}$ epochs.

C. Model-based CoSaMP with additional positivity constraint

The CoSaMP algorithm (Needell & Tropp, 2009) is a simple iterative and greedy algorithm used to recover a K-sparse vector from the linear measurements. The model-based CoSaMP algorithm (Algorithm 1 of (Baraniuk et al., 2010)) is a modification of the CoSaMP algorithm. It uses the prior knowledge about the support of the K-sparse vector, which is assumed to follow a predefined *structured sparsity model*. In this section we slightly modify the model-based CoSaMP algorithm to ensure that the output vector follows the given sparsity model and is also *nonnegative*.

To present the pseudocode, we need a few definitions. We begin with a formal definition for the structured sparsity model $\mathcal{M}_K$ and the sparse approximation algorithm $\mathbb{M}$. For a vector $x \in \mathbb{R}^d$, let $x|_\Omega \in \mathbb{R}^{|\Omega|}$ be entries of x in the index set $\Omega \in [d]$. Let $\Omega^C = [d] - \Omega$ be the complement of set Ω.

Definition 1 ((Baraniuk et al., 2010))**.** *A structured sparsity model $\mathcal{M}_K$ is defined as the union of m_K canonical K-dimensional subspaces*

$$\mathcal{M}_K = \bigcup_{m=1}^{m_K} \mathcal{X}_m \quad \text{s.t. } \mathcal{X}_m = \{x : x|_{\Omega_m} \in \mathbb{R}^K, x|_{\Omega_m^C} = 0\}, \tag{23}$$

where $\{\Omega_1, ..., \Omega_{m_K}\}$ is the set containing all allowed supports, with $|\Omega_m| = K$ for each $m = 1, ..., m_K$, and each subspace $\mathcal{X}_m$ contains all signals x with supp$(x) \subset \Omega_m$.

We define $\mathbb{M}(x, K)$ as the algorithm that obtains the best K-term structured sparse approximation of x in the union of subspaces $\mathcal{M}_K$:

$$\mathbb{M}(x, K) = \arg\min_{\bar{x}\in\mathcal{M}_K} \|x - \bar{x}\|_2. \tag{24}$$

Dataset	Depth	Batch size	Learning rate	$N_{\max}$	$N_{\text{validation}}$	$N_{\text{no improve}}$
Toy	10	128	0.01	2e4	10	5
Synthetic1	10	128	0.01	2e4	10	5
Synthetic2	5	128	0.01	2e4	10	1
Synthetic3	5	128	0.01	2e4	10	1
Amazon	60	256	0.01	2e4	1	1
Wiki10-31K	10	256	0.001	5e3	10	1
RCV1	10	256	0.001	1e3	1	50

Table 4. Training parameters.

We next define an enlarged set of subspaces $\mathcal{M}_K^B$ and the associated sparse approximation algorithm.

Definition 2 ((Baraniuk et al., 2010)). *The B-order sum for the set $\mathcal{M}_K$, with $B > 1$ an integer, is defined as*

$$\mathcal{M}_K^B = \left\{ \sum_{r=1}^{B} x^{(r)}, \quad \text{with } x^{(r)} \in \mathcal{M}_K \right\}. \tag{25}$$

We define $\mathbb{M}_B(x, K)$ as the algorithm that obtains the best approximation of x in the union of subspaces $\mathcal{M}_K^B$:

$$\mathbb{M}_B(x, K) = \arg \min_{\bar{x} \in \mathcal{M}_K^B} \|x - \bar{x}\|_2. \tag{26}$$

Algorithm 1 presents the model-based CoSaMP with positivity constraint. Comparing Algorithm 1 with the original model-based CoSaMP algorithm (Algorithm 1 of (Baraniuk et al., 2010)), we note that the only different is that Algorithm 1 has an extra step (Step 6). In Step 6 we take a ReLU operation on b to ensure that $\hat{x}_i$ is always nonnegative after Step 7.

We now show that Algorithm 1 has the same performance guarantee as the original model-based CoSaMP algorithm for structured sparse signals. Speficially, we will show that Theorem 4 of (Baraniuk et al., 2010) also applies to Algorithm 1. In (Baraniuk et al., 2010), the proof of Theorem 4 is based on six lemmas (Appendix D), among which the only lemma that is related to Step 6-7 is Lemma 6. It then suffices to prove that this lemma is also true for Algorithm 1 under the constraint that the true vector x is nonnegative.

Lemma (Prunning). *The pruned approximation $\hat{x}_i = \mathbb{M}(\hat{b}, K)$ is such that*

$$\|x - \hat{x}_i\|_2 \leq 2\|x - b\|_2. \tag{27}$$

Proof. Since $\hat{x}_i$ is the K-best approximation of $\hat{b}$ in $\mathcal{M}_K$, and $x \in \mathcal{M}_K$, we have

$$\|x - \hat{x}_i\|_2 \leq \|x - \hat{b}\|_2 + \|\hat{b} - \hat{x}_i\|_2 \leq 2\|x - \hat{b}\|_2 \leq 2\|x - b\|_2, \tag{28}$$

where the last inequality follows from that $\hat{b} = \max\{0, b\}$, and $x \geq 0$. □

The above lemma matches Lemma 6, which is used to prove Theorem 4 in (Baraniuk et al., 2010). Since the other lemmas (i.e., Lemma 1-5 in Appendix D of (Baraniuk et al., 2010)) still hold for Algorithm 1, we conclude that the performance guarantee for structured sparse signals (i.e., Theorem 4 of (Baraniuk et al., 2010)) is also true for Algorithm 1.

In Figure 4, we compare the recovery performance of two decoding algorithms: 1) model-based CoSaMP algorithm (Algorithm 1 of (Baraniuk et al., 2010)) and 2) model-based CoSaMP algorithm with positivity constraint (indicated by "Model-based CoSaMP pos" in Figure 4). We use random Gaussian matrices as the measurement matrices. Since our sparse datasets are all nonnegative, adding the positivity constraint to the decoding algorithm is able to improve the recovery performance.

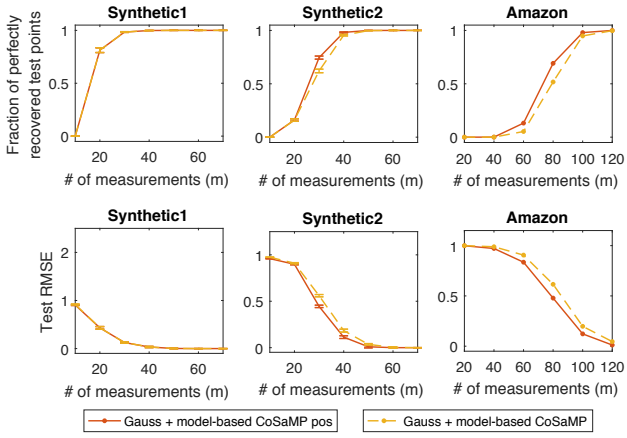

Figure 4. Incorporating the positivity constraint to the model-based CoSaMP algorithm improves its recovery performance.

D. Additional experimental results

D.1. A toy experiment

We use a simple example to illustrate that the measurement matrix learned from our autoencoder is adapted to the training samples. The toy dataset is generated as follows: each

Algorithm 1 Model-based CoSaMP with positivity constraint

Inputs: measurement matrix A, measurements y, structured sparse approximation algorithm $\mathbb{M}$
Output: K-sparse approximation $\hat{x}$ to the true signal x, which is assumed to be nonnegative
$\hat{x}_0 = 0$, $r = y$; $i = 0$ {initialize}
while halting criterion false **do**
1. $i \leftarrow i + 1$
2. $e \leftarrow A^T r$ {form signal residual estimate}
3. $\Omega \leftarrow \text{supp}(\mathbb{M}_2(e, K))$ {prune residual estimate according to structure}
4. $T \leftarrow \Omega \cup \text{supp}(\hat{x}_{i-1})$ {merge supports}
5. $b|_T \leftarrow A_T^\dagger y$, $b|_{T^C} \leftarrow 0$ {form signal estimate by least-squares}
6. $\hat{b} = \max\{0, b\}$ {set the negative entries to be zero}
7. $\hat{x}_i \leftarrow \mathbb{M}(\hat{b}, K)$ {prune signal estimate according to structure}
8. $r \leftarrow y - A\hat{x}_i$ {update measurement residual}

end while
return $\hat{x} \leftarrow \hat{x}_i$

vector $x \in \mathbb{R}^{100}$ has 5 nonzeros randomly located in the first 20 dimensions; the nonzeros are random values between [0,1]. We train ℓ_1-AE on a training set with 6000 samples. The parameters are $T = 10$, $m = 10$, and learning rate 0.01. A validation set with 2000 samples is used for early-stopping.After training, we plot the matrix A in Figure 5. The entries with large values are concentrated in the first 20 dimensions. This agrees with the specific structure in the toy dataset.

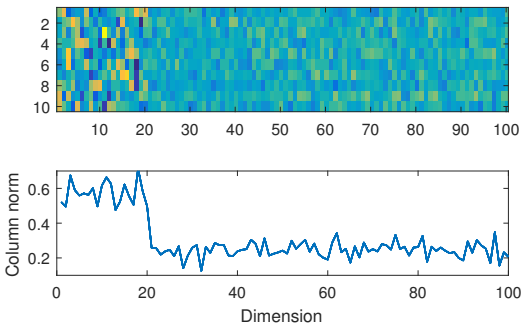

Figure 5. Visualization of the learned matrix $A \in \mathbb{R}^{10\times 100}$ on the toy dataset: a color map of the matrix (upper), the column-wise ℓ_2 norm (lower). Every sample in the toy dataset has 5 nonzeros, located randomly in the first 20 dimensions.

D.2. Random partial Fourier matrices

Figure 6 is a counterpart of Figure 2. The only difference is that in Figure 6 we use random partial Fourier matrices in place of random Gaussian matrices. A random $M \times N$ partial Fourier matrix is obtained by choosing M rows uniformly and independently with replacement from the $N \times N$ discrete Fourier transform (DFT) matrix. We then scale each entry to have absolute value $1/\sqrt{M}$ (Haviv & Regev, 2017). Because the DFT matrix is complex, to obtain m *real* measurements, we draw $m/2$ random rows from a DFT matrix to form the partial Fourier matrix.

A random partial Fourier matrix is a Vandermonde matrix. According to (Donoho & Tanner, 2005), one can exactly recover a k-sparse nonnegative vector from $2k$ measurements using a Vandermonde matrix (Donoho & Tanner, 2005). However, the Vandermonde matrices are numerically unstable in practice (Pan, 2016), which is consistent with our empirical observation. Comparing Figure 6 with Figure 2, we see that the recovery performance of a random partial Fourier matrix has larger variance than that of a random Gaussian matrix.

D.3. Precision score comparisons for extreme multi-label learning

Table 5 compares the precision scores (P@1, P@3, P@5) over two benchmark datasets. For SLEEC, the precision scores we obtained by running their code (and combining 5 models in the ensemble) are consistent with those reported in the benchmark website (Bhatia et al., 2017). Compared to SLEEC, our method (which learns label embeddings via training an autoencoder ℓ_1-AE) is able to achieve better or comparable precision scores. For our method, we have experimented with three prediction approaches (denoted as "ℓ_1-AE 1/2/3" in Table 5): 1) use the nearest neighbor method (same as SLEEC); 2) use the decoder of the trained ℓ_1-AE (which maps from the embedding space to label space); 3) use an average of the label vectors obtained from 1) and 2). As indicated in Table 5, the third prediction approach performs the best.

D.4. ℓ_1-minimization with positivity constraint

We compare the recovery performance between solving an ℓ_1-min (4) and an ℓ_1-min with positivity constraint (15).

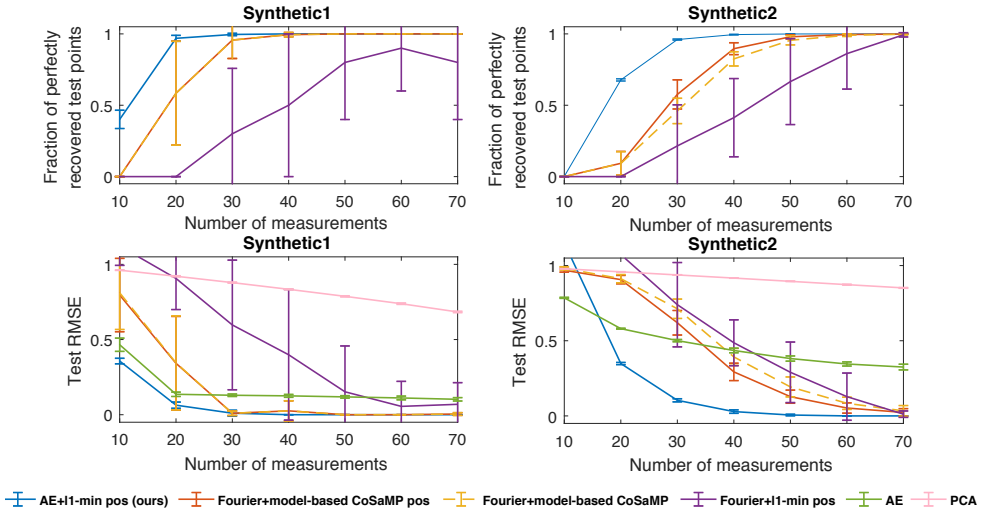

Figure 6. Recovery performance of random partial Fourier matrices. Best viewed in color. Similar to Figure 2, the error bars represent the standard deviation across 10 randomly generated datasets. We see that the recovery performance of a random partial Fourier matrix (shown in this figure) has a larger variance than that of a random Gaussian matrix (shown in Figure 2).

The results are shown in Figure 7. We experiment with two measurement matrices: 1) the one obtained from training our autoencoder, and 2) random Gaussian matrices. As shown in Figure 7, adding a positivity constraint to the ℓ_1-minimization improves the recovery performance for nonnegative input vectors.

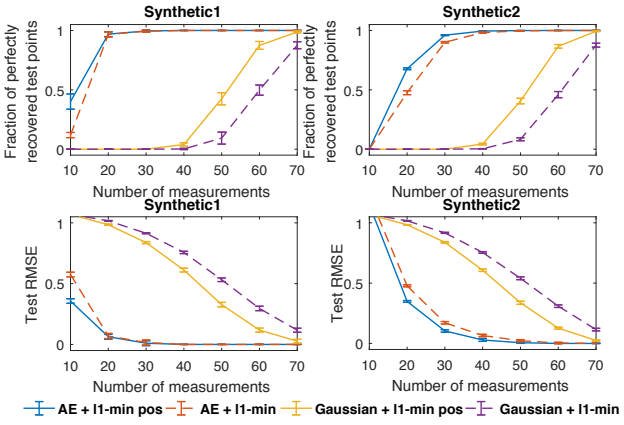

Figure 7. A comparison of the recovery performance between ℓ_1-min (4) and the ℓ_1-min with positivity constraint (15). The sparse recovery performance is measured on the test set. Best viewed in color. We plot the mean and standard deviation (indicated by the error bars) across 10 randomly generated datasets. Adding a positivity constraint to the ℓ_1-minimization gives better recovery performance than a vanilla ℓ_1-minimization.

D.5. Singular values of the learned measurement matrices

We have shown that the measurement matrix obtained from training our autoencoder is able to capture the sparsity structure of the training data. We are now interested in looking at those data-dependent measurement matrices more closely. Table 6 shows that those matrices have singular values close to one. Recall that in Section 3.1 we show that matrices with all singular values being ones have a simple form for the projected subgradient update (12). Our decoder is designed based on this simple update rule. Although we do not explicitly enforce this constraint during training, Table 6 indicates that the learned matrices are not far from the constraint set.

D.6. Synthetic data with no extra structure

We conducted an experiment on a synthetic dataset with no extra structure. Every sample has dimension 1000 and 10 non-zeros, the support of which is randomly selected from all possible support sets. The training/validation/test set contains 6000/2000/2000 random vectors. As shown in Table 7, the learned measurement matrix has similar performance as a random Gaussian measurement matrix.

D.7. Autoencoder with unrolled ISTA

Our autoencoder ℓ_1-AE is designed by unrolling the projected subgradient algorithm of the standard ℓ_1-minimization decoder. We can indeed design a different autoencoder by unrolling other algorithms. One option is the ISTA (Iterative Shrinkage-Thresholding Algorithm) algorithm of the LASSO problem. Comparing the performance of those autoencoders is definitely an interesting

Table 5. Comparisons of precision scores: P@1, P@3, P@5.

Dataset	EURLex-4K			Wiki10-31K		
# models in the ensemble	1	3	5	1	3	5
SLEEC	0.7600	0.7900	0.7944	0.8356	0.8603	0.8600
ℓ_1-AE 1	0.7655	0.7928	0.7931	0.8529	0.8564	0.8597
ℓ_1-AE 2	0.7949	0.8033	0.8070	0.8560	0.8579	0.8583
ℓ_1-AE 3	**0.8062**	**0.8151**	**0.8136**	**0.8617**	**0.8640**	**0.8630**

Dataset	EURLex-4K			Wiki10-31K		
# models in the ensenble	1	3	5	1	3	5
SLEEC	0.6116	0.6403	0.6444	0.7046	0.7304	0.7357
ℓ_1-AE 1	0.6094	0.0.6347	0.6360	0.7230	0.7298	0.7323
ℓ_1-AE 2	0.6284	0.6489	0.6575	0.7262	0.7293	0.7296
ℓ_1-AE 3	**0.6500**	**0.6671**	**0.6693**	**0.7361**	**0.7367**	**0.7373**

Dataset	EURLex-4K			Wiki10-31K		
# models in the ensemble	1	3	5	1	3	5
SLEEC	0.4965	0.5214	0.5275	0.5979	0.6286	0.6311
ℓ_1-AE 1	0.4966	0.5154	0.5209	0.6135	0.6198	0.6230
ℓ_1-AE 2	0.5053	0.5315	0.5421	0.6175	0.6245	0.6268
ℓ_1-AE 3	**0.5353**	**0.5515**	**0.5549**	**0.6290**	**0.6322**	**0.6341**

Dataset	σ_{largest}	σ_{smallest}
Synthetic1	1.117 ± 0.003	0.789 ± 0.214
Synthetic2	1.113 ± 0.006	0.929 ± 0.259
Synthetic3	1.162 ± 0.014	0.927 ± 0.141
Amazon	1.040 ± 0.021	0.804 ± 0.039
Wiki10-31K	1.097 ± 0.003	0.899 ± 0.044
RCV1	1.063 ± 0.016	0.784 ± 0.034

Table 6. Range of the singular values of the measurement matrices $A \in \mathbb{R}^{m \times d}$ obtained from training ℓ_1-AE . The mean and standard deviation is computed by varying the number of m (i.e., the "number of measurements" in Figure 2).

# measurements	30	50	70
ℓ_1-AE + ℓ_1-min pos	0.8084	0.1901	0.0016
Gaussian + ℓ_1-min pos	0.8187	0.1955	0.0003

Table 7. Comparison of test RMSE on a synthetic dataset with no extra structure.

direction for future work. We have performed some initial experiments on this. We designed a new autoencoder ISTA-AE with a linear encoder and a nonlinear decoder by unrolling the ISTA algrithm. On the Synthetic1 dataset, ℓ_1-AE performed better than LISTA (we unrolled ten steps for both decoders): with 10 measurements, the test RMSEs are 0.894 (ISTA-AE), 0.795 (ISTA-AE + ℓ_1-min pos), 0.465 (ℓ_1-AE) and 0.357 (ℓ_1-AE + ℓ_1-min pos).

D.8. Additional experiments of LBCS

We experimented with four variations of LBCS: two different basis matrices (random Gaussian matrix and DCT matrix), two different decoders (ℓ_1-minimization and linear decoder). As shown in Figure 8, the combination of Gaussian and ℓ_1-minimization performs the best.

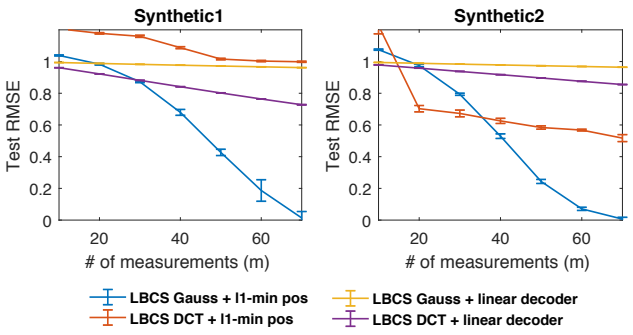

Figure 8. We compare four variations of the LBCS method proposed in (Baldassarre et al., 2016; Li & Cevher, 2016): two basis matrices (random Gaussian and DCT matrix); two decoders (ℓ_1-minimization and linear decoding). The combination of "Gaussian + ℓ_1-minimization" performs the best. Best viewed in color. For each method, we plot the mean and standard deviation (indicated by the error bars) across 10 randomly generated datasets.