DeepPursuit: Uniting Classical Wisdom and Deep RL for Sparse Recovery

Ziheng Chen*, Sichen Zhong†, Jianshu Chen‡ and Yue Zhao§*
*Department of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY
†Splunk Inc., San Francisco, CA
‡Tencent AI Lab, Bellevue, WA
§Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY

***Abstract*—In this paper, we formulate sparse signal recovery as a sequential decision making problem (modeled by Markov Decision Processes). Based on the formulation, we propose *DeepPursuit*, a novel sparse recovery algorithm that learns to recover sparse signals via deep reinforcement learning (RL) and Monte Carlo Tree Search (MCTS). To substantially enhance the learning speed and performance, DeepPursuit (i) employs a novel residual-type policy/value network architecture that organically incorporates the classical wisdom from the Orthogonal Matching Pursuit (OMP) algorithm, and (ii) exploits the available ground-truth knowledge to guide the MCTS during the training process. Experimental results for general random sparse signal recovery demonstrate that, with very low computational complexity, the DeepPursuit algorithm significantly outperforms the state-of-the-art algorithms. Even higher performance gains are observed with experiments on the MNIST dataset.**

I. Introduction

We consider the compressed sensing (CS) problem [1]–[4], where for a given measurement matrix $A \in \mathbb{R}^{m\times n}$, $m \ll n$, and a (noiseless) observation vector $y = Ax_0$, we want to recover a k-sparse vector/signal x_0 ($k < m$)[1]. Formally, it can be formulated as:

$$\underset{x}{\text{minimize}} \quad \|x\|_0 \quad \text{subject to} \quad Ax = y, \tag{1}$$

where the ℓ_0-norm $\|\cdot\|_0$ of a vector is defined as its number of non-zero values. Notably, solving (1) entails minimizing an ℓ_0 norm and is an NP-hard problem. The readers are referred to the seminal papers [1], [2] for a comprehensive survey of the compressed sensing problem. Compressed sensing and sparse signal recovery have seen wide applications in many areas including image processing [5], magnetic resonance imaging (MRI) [4], and seismology [6] among others.

It has been well understood under what conditions solving the NP-hard ℓ_0-minimization problem (1) guarantees to recover the correct sparse signal x_0 [3]. On the other hand, there have been many studies on analyzing conditions under which *efficient algorithms* guarantee to solve (1) *with high probabilities*. In particular, the restricted isometry property (RIP) of A allows the solution of an ℓ_1-minimization problem to recover the correct sparse signal x_0 [2]. There is however a significant gap between the ℓ_0 and ℓ_1 minimization based conditions. Even if the provable RIP conditions are not satisfied, empirical evidence demonstrates that, efficient algorithms can still sometimes achieve good performance in the recovery of x_0 [7], [8]. As such, there is an enormous potential in developing novel computationally efficient algorithms to approach ideally solving the NP-hard ℓ_0 minimization problem. Such algorithmic improvement will lead to significant reduction in the number of measurements needed for signal recovery.

Related Work Efficient algorithms for solving the sparse recovery problem (1) have been extensively studied, including convex relaxation [3], [9], matching and subspace pursuit [7], [10]–[12], and iterative thresholding [13], among others. Recent advancements in machine learning have opened a new frontier for solving compressed sensing problems, in particular by taking a deep learning approach. The works in [14]–[17] apply DNNs and RNNs for encoding and/or decoding of signals x_0. Modern generative models have also been used to encode signals with strong priors and design the measurement function [18], [19]. The latter is also addressed in [20] using MCTS. Different from the above works, our innovation with machine learning is on signal *recovery algorithms*. Supervised learning approaches for training sequential signal support recovery policies have been developed [21]. While they are effective for recovering signals with strong statistical priors, they are shown to underperform classical algorithms such as OMP for general sparse signals.

In this paper, we formulate sparse signal recovery as a sequential decision making problem, modeled by Markov Decision Processes (MDP), where the signal support of x, or equivalently, the columns of A to represent y are sequentially selected. Based on this formulation, we employ an RL-based framework aided with Monte Carlo Tree Search (MCTS) for training signal recovery policies [22]. We develop *DeepPursuit*, a novel sparse recovery algorithm that (i) employs a novel residual-type policy network architecture that organically incorporates the classical wisdom from the Orthogonal Matching Pursuit (OMP) algorithm [10], and (ii) exploits the available ground-truth knowledge to guide the MCTS during training. As such, the classical wisdom from OMP is effectively integrated with that from Deep RL in a unified learning framework. We conduct experiments to evaluate our proposed DeepPursuit

This work is supported by the National Science Foundation under Grant ECCS-2025152.

[1]We say that a vector is k-sparse if it has k non-zero values.

algorithm on general random sparse signals and compare it to existing state-of-the-art methods. The experimental results demonstrate that DeepPursuit significantly outperforms the state-of-the-art methods in speed and/or performance. We note that, in the testing stage, the DeepPursuit algorithm has a very low computational complexity.

II. Sparse Signal Recovery as a Markov Decision Process

A. MDP Formulation of Sparse Recovery

We formulate the sparse signal recovery problem as a sequential decision making problem. Note that the key to the successful sparse recovery is to choose the correct subset of the columns of A, or equivalently, the *support* of x, such that the optimization problem (1) is solved. Equivalently, this can be reformulated as the following problem. An agent is employed to sequentially choose one column of A at a time until it selects up to k columns: The agent succeeds if the selected columns in the end meet the constraint in (1) and minimize the ℓ_0-loss in (1). Next, we formally define such an MDP problem.

State and Action Spaces A state s is defined to be a pair (y, S), where y is the observed signal generated according to x_0 (via $y = Ax_0$)). Let $S \subseteq [n]$ denotes the set of the already selected column indices of A, where $[n] \triangleq \{1, \ldots, n\}$. Provided that the matrix A is given, the state does not depend on this measurement matrix. We define the terminal states to be the states $s = (y, S)$ that satisfy either: (a) a maximum-considered number of columns have been selected, or (b) $||A_S x_s - y||_2^2 < \epsilon$ for some given ϵ, where A_S denotes the submatrix of A constructed by the selected columns in S, and x_s is the solution to the following least-square solution for the given signal support S:

$$x_s \triangleq \arg\min_z ||A_S z - y||_2^2. \tag{2}$$

The feasible action space at a state $s = (y, S)$ is defined to be $\mathcal{A}_s = [n] \setminus S$; that is, a valid action at state s is any column from the remaining unchosen ones.

Transition When an action a is taken (i.e., a new column a of A is selected) at state $s = (y, S)$, the next state $s' = (y, S \cup \{a\})$ (and hence the MDP transition) is determined and known.

Reward We define the reward to be: $R(s, a, s') := -1 - \beta \left(||A_{S'} x_{s'} - y||_2^2 - ||A_S x_s - y||_2^2 \right)$, where s, s' are the current and next states, and x_s is the least-square solution given by (2) (and similarly for $x_{s'}$). Such a reward design ensures that the cummulative reward when reaching a terminal state s is

$$R^{\text{cum}}(s) = -||x_s||_0 - \beta ||A_S x_s - y||_2^2. \tag{3}$$

As such, this cummulative reward consists of two parts: the first part is the ℓ_0 term that measures the sparsity of the solution, and the second part is the optimal least-square-error given the column choices in S. $\beta > 0$ is a hyperparameter that controls the balance between sparsity and goodness of fit. We note that, the purpose of such a "reward decomposition" is to avoid the potential sparse-reward issue that occurrs when we only have the terminal rewards (3) with no intermediate ones.

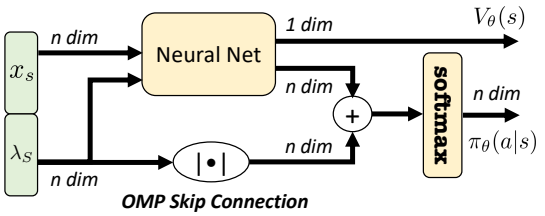

Fig. 1. The OMP-Residual Policy/Value Network.

B. Learning-to-Recover via Reinforcement Learning

Given a measurement matrix A, we have a set of (sparse) signals x_0, and generate observations y according to $y = Ax_0$. Based on these signal-observation pairs $\{(x_0, y)\}$, we use an RL framework to learn a policy $\pi(a|s)$(modeled by a policy network), which sequentially selects the columns of A and reconstructs the sparse signal x_0. The RL objective is to learn the $\pi(a|s)$ that maximizes the cumulative reward, which leads to recovering x to be the same as the ground truth x_0 with high probability. In RL training, MCTS is employed to generate high quality experience to update the policy. After the training, at the testing stage, the learned policy network $\pi(a|s)$ can then recover the sparse signal x_0 *efficiently* for any *unseen* y.

We note that there are existing works using supervised learning approaches (as opposed to RL) to train a sequential decision policy for choosing the signal support, but fail to outperform OMP for general signals without a strong prior [21]. This motivates us to (i) develop the proposed RL approach, (ii) design our method to incorporate the wisdom of OMP and learn beyond the classical algorithms, while still (iii) leveraging the available ground truth knowledge as the supervision and guidance signals.

III. The *DeepPursuit* Algorithm

The key innovations of the proposed DeepPursuit Algorithm stem from two objectives. Firstly, we seek to incorporate a classical sparse recovery algorithm, OMP, into the learning of a policy network. As such, the wisdom from both classical compressed sensing algorithm design and deep reinforcement learning are organically integrated. Secondly, we seek to leverage the fact that the ground truth signals x_0 are available at the training stage, and use them to effectively guide the training of a policy network. We achieve the first objective by designing a novel *OMP-Residual Policy Network (OMP-ResNet)*, and the second objective by incorporating the ground-truth-knowledge to guide the MCTS process at the training stage.

A. OMP-Residual Policy/Value Network

To learn a policy in the sequential decision making formulation of sparse signal recovery (cf. Section II-A), we employ a single neural network to jointly model the policy $\pi_\theta(a|s)$ and the state-value function $V_\theta(s)$. The policy $\pi_\theta(a|s)$ defines a probability distribution over all actions for a given state s, where the action set includes the possible next columns of A to pick and a stopping action. The input of the policy/value

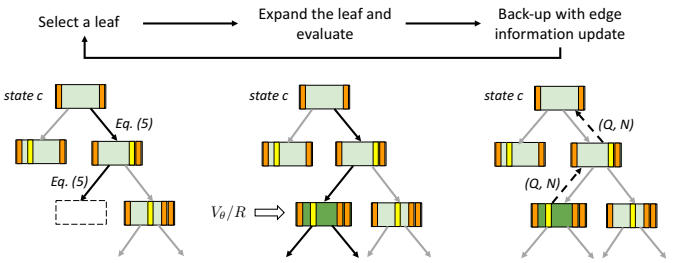

Fig. 2. MCTS rollouts for sparse recovery.

network consists of two parts: (x_s, λ_s). The first part is the least-square solution in (2) extended to an n-dimensional vector with zeros padded in the positions not in S. The second part is given by

$$\lambda_s := A^T(y - A_S x_s) \in \mathbb{R}^n,$$

where $y - A_S x_s$ is the least-square's residue. As will be shown next, having λ_s as an input feature allows us to include an "OMP skip connection" in the policy network architecture.

In desigining the architecture of the policy network, we seek to leverage OMP which a) is widely accepted as a very fast and effective heuristic for sparse recovery, and b) shares the same sequential decision making nature as our MDP formulation. Notably, the OMP algorithm sequentially chooses the column index whose corresponding component in $|\lambda_s|$ is the largest, where $|\lambda_s|$ captures the correlation between the columns of A and the least-square's residue. Such a *"hard-max"* decision policy can be approximated by $\texttt{Softmax}(|\lambda_s|)$. On the other hand, the policy network also employs a softmax layer, with a log-probability vector as its input. Thus, a natural way of combining the decision of OMP and that of a trained policy network is to element-wise *add* the logits from both parts, meaning that the respective probabilities are multiplied.

Based on the above, we design a policy network architecture as in Figure 1. The upper branch is a general neural network, whereas the lower branch is an "OMP skip connection" that implements $|\lambda_s|$ to mimic the "OMP-policy". Adding the OMP skip connection enables us to learn beyond what OMP can do. In other words, we free the upper branch from learning what OMP already does, and provide it a "head start" as it only needs to focus on what OMP cannot do. Since it shares a high-level intuition with the residual network [23], we call our new architecture as an "OMP-Residual Policy Network".

B. Knowledge-Guided Monte Carlo Tree Search

Since the MDP transition is deterministic and known, we are able to use such model-based information to perform planning with MCTS in RL training [24]. In particular, we employ the general algorithmic framework of AlphaZero [24], and introduce two novel components that incorporate supervision signals into the MCTS and RL process (cf. Figure 4).

1) Guiding the MCTS with Ground-Truth Knowledge: The key search decision during the MCTS rollouts is how to select an action a_t at each state s_t experienced. On the one hand, we incorporate the decision framework of PUCT (Polynomial Upper Confidence bound for Trees) [24]. On the other hand, we leverage the availability of the ground truths x_0 in the generated training data as a supervision signal to guide the search. Specifically, we introduce a perturbation vector ηe_{a_0} in the PUCT framework as follows:

$$\tilde{\pi}_\theta(a|s_t) \propto \pi_\theta(a|s_t) + \eta e_{a_0}, \quad \sum_a \tilde{\pi}_\theta(a|s_t) = 1, \tag{4}$$

$$a_t = \arg\max_a \left\{ Q(s_t, a) + c_{\text{puct}} \cdot \tilde{\pi}_\theta(a|s_t) \frac{\sqrt{\sum_b N(s_t, b)}}{N(s_t, a) + 1} \right\}, \tag{5}$$

where, in (4), a) s_t denotes the state at step t during the MCTS simulations; b) $\pi_\theta(a|s_t)$ is the output (action probabilities) of the policy network; c) e_{a_0} is a 1-hot vector at a position a_0 randomly selected within the *ground truth* signal support; d) $\eta > 0$ is a hyperparameter that controls the influence of this ground truth guidance; and e) $\tilde{\pi}_\theta(a|s_t)$ is the action probabilities with such guidance. $Q(s_t, a)$ is the action value function, and $N(s_t, a)$ is the visiting count. (5) is PUCT with c_{puct} being a hyper-parameter that controls the tradeoff between exploration and exploitation.

As such, during the MCTS process, the above guidance makes it more likely to select a support among the ground truth labels. By varying η, we balance between exploring more generally and learning primarily from the ground truth.

2) Training with a Diminishing OMP-bias: After the (pseudo) empirical probability labels are computed from MCTS, denoted by $p^M \in \mathbb{R}^n$, we further introduce a bias in these labels with the knowledge from OMP. Specifically, as OMP produces a deterministic choice, we encode this choice in a 1-hot vector, $p^O \in \mathbb{R}^n$, whose value is one at the OMP's choice and zero elsewhere. We then bias the (pseudo) label p^M from MCTS with p^O in the cross entropy loss:

$$l = -[(1 - \mu)p^M + \mu p^O] \log \pi_\theta(s), \tag{6}$$

where μ is a hyperparameter that controls the contributions from both labels, and $\pi_\theta(s) \in \mathbb{R}^n$ is the vector of all the action probabilities from the policy network. This step of biasing the labels by OMP is in fact designed in conjunction with the OMP skip connection. The rationale is that, at the initial stage of training, the (pseudo) labels from MCTS have poor qualities, and biasing them with the OMP's choice improves the label quality. Next, as training progresses and the policy network becomes better, we anneal this bias by gradually reducing μ.

IV. Simulation

In this section, we present experimental results for evaluating the proposed DeepPursuit algorithm with (i) general random sparse signals, and (ii) the MNIST dataset [25]. In all the experiments, the upper branch of the OMP-ResNet in Figure 1 uses a neural network with two hidden layers and ReLU

TABLE I
VECTOR RECOVERY ACCURACY (HIGHER IS BETTER) OF DEEPPURSUIT VS. EXISTING ALGORITHMS.

Matrix Size	Sparse Recovery Alg.	Accuracy (%) for various sparsity k				Testing time (millisecond)
		$k = 2$ (20)	$k = 3$	$k = 4$	$k = 5$	
10×100	BP (ℓ_1-min)	77.6	32.5	6.0	0.8	20.0
	OMP	79.6	41.3	11.0	2.0	0.49
	CoSaMP [11]	74.7	28.9	3.5	0.1	1.89
	Subspace Pursuit [12]	77.4	36.9	8.8	1.7	0.76
	DeepPursuit (ours)	$\mathbf{83.9 \pm 1.2}$	$\mathbf{47.5 \pm 0.7}$	$\mathbf{13.6 \pm 1.1}$	$\mathbf{2.2 \pm 0.0}$	0.67
15×150	BP (ℓ_1-min)	**97.2**	61.0	26.5	7.3	90.0
	OMP	94.3	75.6	46.3	22.0	0.52
	CoSaMP [11]	93.4	71.1	31.2	7.8	2.01
	Subspace Pursuit [12]	96.8	73.8	40.5	19.5	0.90
	DeepPursuit (ours)	94.3 ± 0.6	$\mathbf{79.8 \pm 1.3}$	$\mathbf{55.1 \pm 1.6}$	$\mathbf{25.6 \pm 1.1}$	0.86
20×250	BP (ℓ_1-min)	**99**	87	52	26.9	170.0
	OMP	96.5	86.1	67.5	40.8	0.89
	CoSaMP [11]	98.6	85.5	60.8	31.1	2.79
	Subspace Pursuit [12]	99.5	**87.2**	68.7	39.8	1.35
	DeepPursuit (ours)	96.3 ± 0.5	86.9 ± 1.1	$\mathbf{71.2 \pm 1.2}$	$\mathbf{43.6 \pm 1}$	0.96

activations followed by two separate output heads that models $\pi_\theta(a|s)$ and $V_\theta(s)$, respectively.

A. General Random Sparse Signal Recovery

Training Data Generation We conduct experiments on four measurement matrices of sizes 10×100, 15×150, 20×250 and 200×600, respectively. Each matrix A is generated with entries sampled from an independent and identically distributed (i.i.d) standard Gaussian $\mathcal{N}(0, 1)$ distribution. In each iteration (i.e., between consecutive updates of the policy/value network) in the training process, 1600 i.i.d. random samples of x_0 are generated: a) x_0's sparsity k is randomly generated, b) the locations of the k nonzero elements in x_0 are chosen uniformly at random, and c) the values of the nonzero elements are generated i.i.d. from $U[0, 1]$. $y_0 = Ax_0$ is computed for each x_0, resulting in a (y, x_0) pair. MCTS is then performed on each of these 1600 (y, x_0) pairs.

Testing Data Generation and Evaluation Metric The test data contain (y, x_0) pairs generated i.i.d. in the same way as in training. For each sparsity level k that we evaluate for, we generate 1000 k-sparse test signals x_0 and compute the corresponding y. With $\hat{x}$ as the predicted sparse vector, we define a successful recovery of x_0 as *exactly* satisfying $\hat{x} = x_0$.

Experiment Results We evaluate the testing performance of DeepPursuit and several baseline algorithms. We summarize the main results (along with their per-sample testing times[2]) in Table I and Figure 3. We note that all the performance of DeepPursuit are achieved *without any use of MCTS during testing*, i.e., only by querying the OMP-ResNet. We observe a significant improvement in the recovery success rates of DeepPursuit over the existing algorithms. Moreover, DeepPursuit is only slightly slower than OMP, and orders of magnitude faster than Basis Pursuit (BP, i.e., ℓ_1-minimization).

[2]The testing time is measured on a computer with 5 CPU cores and 1 Nvidia 1080TI GPU.

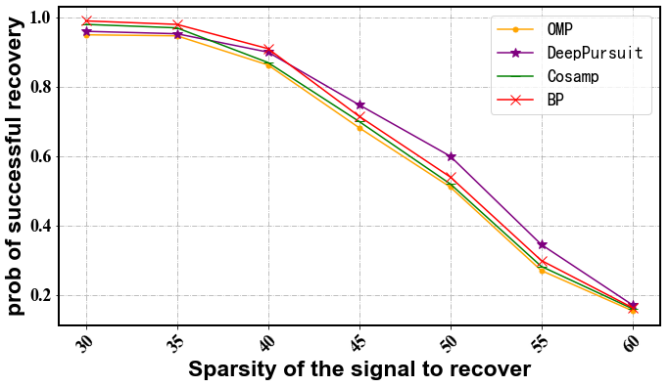

Fig. 3. Vector recovery accuracy for the 200×600 matrix.

Ablation Study We then perform ablation study of DeepPursuit to examine the contribution of the individual components (Table II), which shows that all the components developed in Section III are essential. Note that the results in Tables I and II are performed with 400 MCTS rollouts in training, and none in testing. To understand the importance of MCTS, the performance of DeepPursuit under different numbers of MCTS rollouts in training is depicted in Figures 4. We can see the performance improves significantly as the number of rollouts increase. The gain of employing more rollouts beyond 400 would however be very small, and is hence not quite worth the corresponding extra training time. Generally, it is clear that MCTS plays a key role in improving the training efficiency.

B. Image Recovery with Compressed Measurements

Each image in the MNIST dataset is of size 24×24. We limit the number of nonzero values in each image to 80. We divide each image into four 12×12 blocks (cf. [26]), resulting

TABLE II
ABLATION STUDY OF DEEPPURSUIT ON 15×150 MATRIX A (IN VECTOR RECOVERY ACCURACY (%)).

Method	$k = 2$	$k = 3$	$k = 4$	$k = 5$
DeepPursuit	$\mathbf{94.3 \pm 0.6}$	$\mathbf{79.8 \pm 1.3}$	$\mathbf{55.1 \pm 1.6}$	$\mathbf{25.6 \pm 1.1}$
DeepPursuit w/o OMP skip connection	92.7 ± 0.6	78.6 ± 0.8	47.9 ± 1.0	20.9 ± 1.2
DeepPursuit w/o OMP-biased labels	85.0 ± 6.5	64.1 ± 5.1	23.0 ± 2.0	5.6 ± 0.8
DeepPursuit w/o ground-truth guidance	83.2 ± 6.0	63.1 ± 3.6	26.2 ± 3.6	4.8 ± 1.6
DeepPursuit w/ MCTS within ground-truth only	62.0 ± 6.0	37.0 ± 3.0	18.0 ± 4.2	3.6 ± 1.3

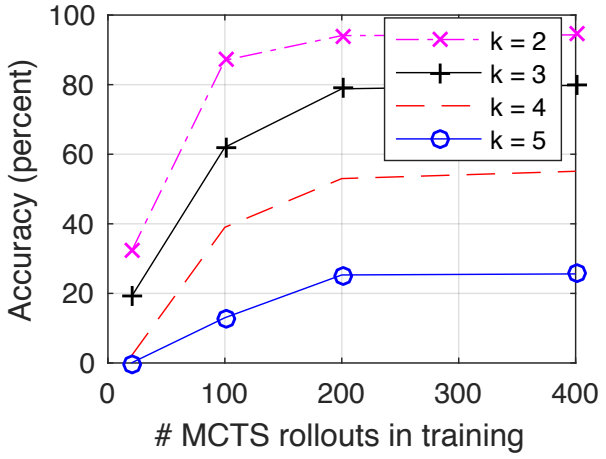

Fig. 4. Testing performance vs. number of MCTS rollouts in training.

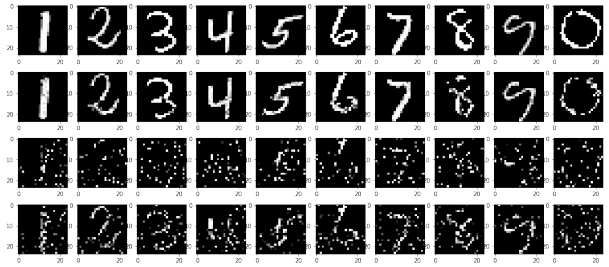

Fig. 5. MNIST image recovery. From top to bottom: original, DeepPursuit, OMP, BP.

Method	DeepPursuit	DeepPursuit $\cup$ OMP	BP	OMP
$\frac{\lvert S \cap S_0 \rvert}{\lvert S_0 \rvert}$	89.0 ± 2.0	89.0 ± 2.1	57.5	53.0

TABLE III
TESTING PERFORMANCE (%) ON THE MNIST DATASET

in a signal dimension of 144. A measurement matrix of size 50×144 is generated with i.i.d. standard Gaussian $N(0, 1)$ distribution. $43,000$ original images are used in the training process, and $1,000$ for testing. We evaluate the following performance metric for image recovery: $|S \cap S_0|/|S_0|$, where $|S_0|$ is the number of nonzero pixels in the original image, and $|S \cap S_0|$ is the number of pixels where both the original and the recovered images are non-zero (i.e., the size of the overlap). We also plot the recovered images.

We summarize the testing performance in Table III, which shows that DeepPursuit significantly outperforms BP and OMP. Next, we plot examples of the recovered images from all the tested algorithms, as well as the original images, in Figure 5. We observe that the recoveries by DeepPursuit are perceptually very close to the original images, whereas those of BP and OMP suffer from major perceptual distortions.

V. CONCLUSION

We have developed DeepPursuit, a new sparse signal recovery algorithm trained under a reinforcement learning (RL) framework. We design a novel residual network architecture that organically integrates the Orthogonal Matching Pursuit algorithm into the learning of a policy/value network. We further employ MCTS in the RL training process, and leverage the supervision signals from the ground-truth knowledge available in the training data to guide the MCTS. We demonstrate that DeepPursuit significantly outperforms the state-of-the-art algorithms for recovering general random sparse signals. Furthermore, we show that DeepPursuit achieves even larger performance gain over OMP and BP on image recovery tasks.

REFERENCES

[1] David L Donoho, "Compressed sensing," *IEEE Trans. on Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[2] Emmanuel J Candes, "The restricted isometry property and its implications for compressed sensing," *Comptes rendus mathematique*, vol. 346, no. 9-10, pp. 589–592, 2008.

[3] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, Dec 2005.

[4] Michael Lustig, David Donoho, and John M Pauly, "Sparse MRI: The application of compressed sensing for rapid mr imaging," *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, vol. 58, no. 6, pp. 1182–1195, 2007.

[5] Yuejie Chi, Louis L Scharf, Ali Pezeshki, and A Robert Calderbank, "Sensitivity to basis mismatch in compressed sensing," *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2182–2195, 2011.

[6] Felix J Herrmann, Michael P Friedlander, and Ozgur Yilmaz, "Fighting the curse of dimensionality: Compressive sensing in exploration seismology," *IEEE Signal Processing Magazine*, vol. 29, no. 3, pp. 88–100, 2012.

[7] David Donoho and Jared Tanner, "Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 367, no. 1906, pp. 4273–4293, 2009.

[8] Guillaume Lecué and Shahar Mendelson, "Sparse recovery under weak moment assumptions," *J. Eur. Math. Soc.(JEMS)*, vol. 19, no. 3, pp. 881–904, 2017.

[9] Scott Shaobing Chen, David L Donoho, and Michael A Saunders, "Atomic decomposition by basis pursuit," *SIAM review*, vol. 43, no. 1, pp. 129–159, 2001.

[10] Stéphane G Mallat and Zhifeng Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on signal processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[11] Deanna Needell and Joel A Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," *Applied and computational harmonic analysis*, vol. 26, no. 3, pp. 301–321, 2009.

[12] Wei Dai and Olgica Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Trans. on Inf. Theory*, vol. 55, no. 5, pp. 2230–2249, 2009.

[13] Ingrid Daubechies, Michel Defrise, and Christine De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Comm. on Pure and Applied Math: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 57, no. 11, pp. 1413–1457, 2004.

[14] Ali Mousavi, Ankit B Patel, and Richard G Baraniuk, "A deep learning approach to structured signal recovery," in *53rd Annual Allerton Conference on Communication, Control, and Computing*. IEEE, 2015, pp. 1336–1343.

[15] Ali Mousavi and Richard G Baraniuk, "Learning to invert: Signal recovery via deep convolutional networks," in *Acoustics, Speech and Signal Processing, IEEE International Conference on*. IEEE, 2017, pp. 2272–2276.

[16] Amir Adler, David Boublil, Michael Elad, and Michael Zibulevsky, "A deep learning approach to block-based compressed sensing of images," *arXiv preprint arXiv:1606.01519*, 2016.

[17] Eliya Nachmani, Elad Marciano, Loren Lugosch, Warren J Gross, David Burshtein, and Yair Beery, "Deep learning methods for improved decoding of linear codes," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 119–131, 2018.

[18] Shanshan Wu, Alexandros G Dimakis, Sujay Sanghavi, Felix X Yu, Daniel Holtmann-Rice, Dmitry Storcheus, Afshin Rostamizadeh, and Sanjiv Kumar, "Learning a compressed sensing measurement matrix via gradient unrolling," *arXiv preprint arXiv:1806.10175*, 2018.

[19] Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis, "Compressed sensing using generative models," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 537–546.

[20] Kyong Hwan Jin, Michael Unser, and Kwang Moo Yi, "Self-supervised deep active accelerated MRI," *arXiv:1901.04547*, 2019.

[21] Dany Merhej, Chaouki Diab, Mohamad Khalil, and Rémy Prost, "Embedding prior knowledge within compressed sensing by neural networks," *IEEE transactions on neural networks*, vol. 22, no. 10, pp. 1638–1649, 2011.

[22] Sichen Zhong, Yue Zhao, and Jianshu Chen, "Learning to recover sparse signals," in *57th Annual Allerton Conf. on Comm., Control, and Computing*. IEEE, 2019, pp. 995–1000.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[24] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al., "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[26] Hamid Palangi, Rabab Ward, and Li Deng, "Distributed compressive sensing: A deep learning approach," *IEEE Transactions on Signal Processing*, vol. 64, no. 17, pp. 4504–4518, 2016.