THE LEARNED INEXACT PROJECT GRADIENT DESCENT ALGORITHM

Raja Giryes[#†] *Yonina C. Eldar*[□] *Alex M. Bronstein*[†] *Guillermo Sapiro*[#*]

[#†] School of Electrical Engineering, Tel Aviv University, Tel Aviv, Israel 69978
[□] Electrical Engineering Department, Technion - IIT, Haifa, Israel, 32000
[†] Computer Science Department, Technion - IIT, Haifa, Israel, 32000
[#] Electrical and Computer Engineering Department, Duke University, Durham, NC, 27708

ABSTRACT

Accelerating iterative algorithms for solving inverse problems using neural networks have become a very popular strategy in the recent years. In this work, we propose a theoretical analysis that may provide an explanation for its success. Our theory relies on the usage of inexact projections with the projected gradient descent (PGD) method. It is demonstrated in various problems including image super-resolution.

Index Terms— Inverse Problems, Sparse Representation, Deep Learning, LISTA, Algorithm Acceleration

1. INTRODUCTION

Consider an inverse problem in which we want to recover $\mathbf{x} \in \mathbb{R}^d$ from $\mathbf{y} = \mathbf{M}\mathbf{x}+\mathbf{e}$, where $\mathbf{M} \in \mathbb{R}^{m\times d}$ is the measurement matrix and $\mathbf{e} \in \mathbb{R}^d$ is additive noise. This setup appears in many applications such as image deblurring, super-resolution and more. Often the recovery of $\mathbf{x}$ from $\mathbf{y}$ is an ill-posed problem, e.g., when $\mathbf{M}$ has fewer rows than columns ($m < n$). For good reconsturction, additional assumptions on the structure of $\mathbf{x}$ is required. A popular strategy is to assume that it resides in a low dimensional set $\mathcal{K}$, e.g., sparse vectors [1]. In this case, the minimization problem becomes

$$\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{M}\mathbf{x}\|_2^2 \quad \text{s.t.} \quad \mathbf{x} \in \mathcal{K}. \tag{1}$$

This can be reformulated in an unconstrained form as

$$\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{M}\mathbf{x}\|_2^2 + \lambda f(\mathbf{x}), \tag{2}$$

where λ is a regularization parameter and f is a cost function related to $\mathcal{K}$. For example, for $\mathcal{K}$ the set of k-sparse vectors, we may pick $f(\cdot) = \|\cdot\|_0$ or its convex relaxation $f(\cdot) = \|\cdot\|_1$.

A popular technique for solving (1) and (2) is using iterative programs such as proximal methods [2, 3] that include the iterative shrinkage-thresholding algorithm (ISTA) [4, 5, 6]. As many applications impose a constraint on the number of computations to be performed to recover $\mathbf{x}$, many acceleration techniques have been proposed for proximal methods [4, 5, 7, 8]. A prominent one is learned ISTA (LISTA) [9], which learns a neural network that has the same structure as ISTA but with only several layers. It reaches virtually the same accuracy as the original ISTA using significantly less iterations. This acceleration by neural networks has been used with many other methods and not only ISTA [10, 11, 12, 13].

Contribution. While many acceleration techniques for ISTA were proposed together with a thorough theoretical analysis, the powerful LISTA method was introduced without mathematical justification for its success. Very recently, a connection was drawn between the convergence speed of LISTA and the factorization of the Gram matrix of $\mathbf{M}$ [14], the restricted isometry property (RIP) [15] and vector approximate message passing (VAMP) [16]. Yet, all these results focus mainly on the case of sparse $\mathbf{x}$. Our work is not restricted to this case and applies to general low-dimensional models.

We focus on the projected gradient descent (PGD) algorithm, whose iterations are almost identical to the ones of ISTA but with an orthogonal projection instead of a proximal mapping. An acceleration technique for it is proposed, which is very similar to the one of LISTA, accompanied with theoretical analysis. We demonstrate the PGD acceleration in two cases: recovery of sparse representation with tree structure and image super-resolution using a pair of dictionaries [17]. For brevity, we omit the proofs from this paper. More details appear in a longer version of this paper [18].

2. BACKGROUND

ISTA is an iterative technique for minimizing (2) given by

$$\mathbf{z}_{t+1} = \mathcal{S}_{f,\mu\lambda}\left(\mathbf{z}_t + \mu\mathbf{M}^*(\mathbf{y} - \mathbf{M}\mathbf{z}_t)\right), \tag{3}$$

where $\mathbf{z}_t$ is its outcome at iteration t, μ is a step-size (constant in our work) obeying $\frac{1}{\mu} \geq \|\mathbf{M}\|$ to ensure convergence [4] and $\mathcal{S}_{f,\lambda}(\cdot)$ is a proximal mapping with parameter λ obeying

$$\mathcal{S}_{f,\lambda}(\mathbf{v}) = \text{argmin}_{\mathbf{z}} \frac{1}{2}\|\mathbf{z} - \mathbf{v}\| + \lambda f(\mathbf{z}). \tag{4}$$

*RG is partially supported by GIF grant no. I-2432-406.10/2016 and ERC-StG grant no. 757497 (SPADE). YE is partially supported by the ERC grant no. 646804-ERC-COG-BNYQ. AB is partially supported by ERC-StG RAPID. GS is partially supported by ONR, NSF, NGA, and ARO.

Authorized licensed use limited to: Institute of Science and Technology Austria (IST). Downloaded on September 22,2026 at 10:55:56 UTC from IEEE Xplore. Restrictions apply.

This mapping has a simple form for many functions f, e.g., if $f(\cdot) = \|\cdot\|_1$, it is an element-wise shrinkage function,

$$\mathcal{S}_{\ell_1,\lambda}(\mathbf{v})[i] = \operatorname{sgn}(\mathbf{v}[i]) \max(0, |\mathbf{v}[i]| - \lambda). \quad (5)$$

Thus, the advantage of ISTA is that its iterations require only the application of matrix multiplications followed by a simple non-linear function. Yet, its main drawback is the large number of iterations that is typically required for convergence.

LISTA accelerates ISTA using the neural network:

$$\mathbf{z}_{t+1} = \mathcal{S}_{f,\lambda}\left(\mathbf{A}\mathbf{y} + \mathbf{U}\mathbf{z}_t\right), \quad (6)$$

where $\mathbf{A} \in \mathbb{R}^{d\times m}$, $\mathbf{U} \in \mathbb{R}^{d\times d}$ and λ are learned from training examples by back-propagation with the minimization objective being the ℓ_2-distance between the final ISTA solution and the LISTA one (after T iterations) [9]. Other objectives may be used, e.g., training LISTA to minimize (2) directly [10].

PGD is similar to ISTA but with an orthogonal projection onto $\mathcal{K}$, $\mathcal{P}_\mathcal{K}$, instead of proximal mapping. It reads as

$$\mathbf{z}_{t+1} = \mathcal{P}_\mathcal{K}\left(\mathbf{z}_t + \mu\mathbf{M}^*(\mathbf{y} - \mathbf{M}\mathbf{z}_t)\right), \quad (7)$$

where μ is the step-size. If $\mathcal{K}$ is the ℓ_1-ball then $\mathcal{P}_\mathcal{K}$ is soft thresholding with a value that varies depending on the projected vector [19]. Note the similarity to (5). This similarity is not unique to the ℓ_1-norm case but happens also for other types of f such as the ℓ_0 pseudo-norm and the nuclear norm.

PGD generalizes the iterative hard thresholding (IHT) algorithm, which was developed for $\mathcal{K}$ being the set of sparse vectors [20] and later used to more general sets [21]. Theory for its convergence has been developed in [22] for the set

$$\mathcal{K} = \left\{\mathbf{z} \in \mathbb{R}^d : f(\mathbf{z}) \le f(\mathbf{x})\right\}.$$

Their theory relies on the tangent cone of the function f at $\mathbf{x}$. We provide here a variant of this theory for the noiseless case that relies directly on $\mathcal{K}$ via its Minkowski difference $\mathcal{K} - \mathcal{K} = \{\mathbf{z} - \mathbf{v} : \mathbf{z}, \mathbf{v} \in \mathcal{K}\}$.

Theorem 2.1 *Let $\mathbf{x} \in \mathcal{K}$, $\mathcal{K} \subset \mathbb{R}^d$ be a closed cone, $\mathbf{M} \in \mathbb{R}^{m\times d}$ and $\mathbf{y} = \mathbf{M}\mathbf{x}$. The estimate $\mathbf{z}_t$ of PGD with a projection onto $\mathcal{K}$ (initialized with $\mathbf{z}_0 = 0$) obeys*

$$\|\mathbf{z}_t - \mathbf{x}\| \le \left(\kappa_\mathcal{K}\rho(\mathcal{K})\right)^t \|\mathbf{x}\|, \quad (8)$$

where $\kappa_\mathcal{K} = 1$ if $\mathcal{K}$ is convex and $\kappa_\mathcal{K} = 2$ otherwise, and

$$\rho(\mathcal{K}) = \rho(\mu, \mathbf{M}, \mathcal{K}) = \sup_{\mathbf{u},\mathbf{v}\in(\mathcal{K}-\mathcal{K})\cap\mathbb{B}^d} \mathbf{u}^*\left(\mathbf{I} - \mu\mathbf{M}^*\mathbf{M}\right)\mathbf{v}, \quad (9)$$

is the convergence rate of PGD.

The *Gaussian Mean Width* measures the complexity of $\mathcal{K}$:

$$\omega_\mathcal{K} = E[\sup_{\mathbf{v}\in(\mathcal{K}-\mathcal{K})\cap\mathbb{B}^d} \langle\mathbf{g}, \mathbf{v}\rangle], \quad \mathbf{g} \sim N(0, \mathbf{I}). \quad (10)$$

For example, if $\mathcal{K}$ is the k-sparse set $\omega_\mathcal{K} = O\left(\sqrt{k\log(d/k)}\right)$, and if it has also a tree structure (an entry may be non-zero only if its parent node is non-zero) $\omega_{\hat{\mathcal{K}}} = O(\sqrt{k})$ [23].

The relationship between the PGD convergence rate and the Gaussian mean width has been developed in theorems 2.2 and 2.4 in [22] for the case that $\mathbf{M}$ is random Gaussian matrix. Using their result it can be shown that the smaller $\omega_\mathcal{K}$, the faster the convergence. More specifically, if m is very close to $\omega_\mathcal{K}$, then we may apply PGD with a step-size $\mu = \frac{1}{(\sqrt{d}+\sqrt{m})^2} \simeq \frac{1}{d}$ and have a convergence rate of

$$\rho(\mathcal{K}) = 1 - O\left(\frac{\sqrt{m} - \omega_\mathcal{K}}{m + d}\right). \quad (11)$$

If $\omega_\mathcal{K}$ is smaller than $\sqrt{m}$ by a certain constant factor, then PGD with the step size $\mu \simeq \frac{1}{m}$ obtains improved convergence

$$\rho(\mathcal{K}) = O\left(\frac{\omega_\mathcal{K}}{\sqrt{m}}\right). \quad (12)$$

Note that (11) and (12) set a limit on the minimal m for which PGD iterations converge to $\mathbf{x}$, namely $m = O(\omega_\mathcal{K}^2)$. This implies $m = O(k\log(d/k))$ for k-sparse vectors and $m = O(k)$ if they have a tree structure. The above connection between $\rho(\mathcal{K})$ and $\omega_\mathcal{K}$ is not unique only to random Gaussian matrices. A similar relationship holds for many other types of $\mathbf{M}$ [22].

3. INEXACT PGD (IPGD)

It may happen that $\mathcal{K}$ is too loose for describing $\mathbf{x}$. Selection of a set $\hat{\mathcal{K}}$ that better characterizes $\mathbf{x}$ will lead to a smaller $\omega_{\hat{\mathcal{K}}}$, resulting in faster convergence. This improvement can be very significant; smaller ω both improves the convergence rate and allows using a larger step-size (see (11) and (12)). For example, consider the case of a k-sparse vector $\mathbf{x}$, whose sparsity pattern obeys a tree structure. If we ignore the tree structure and project only onto the k-sparse set $\mathcal{K}$ then $\omega_\mathcal{K} = O(k\log(d/k))$ [24]. Yet, if we add the tree structure and use

$$\hat{\mathcal{K}} = \{\mathbf{z} \in \mathbb{R}^d : \|\mathbf{z}\|_0 \le \|\mathbf{x}\|_0 \,\&\, \mathbf{z} \text{ obeys a tree structure}\} \quad (13)$$

then $\omega_{\hat{\mathcal{K}}} = O(k)$ [23]. As mentioned above, this improvement might be very significant especially when m is very close to $\omega_\mathcal{K}$. Such an approach was taken in the context of model-based compressed sensing [21], where faster convergence is achieved by projecting onto $\hat{\mathcal{K}}$ instead of $\mathcal{K}$.

A difficulty often encountered is that the projection onto $\hat{\mathcal{K}}$, which may even be unknown, is more complex to implement than the projection onto $\mathcal{K}$. The latter is easier to project onto but provides a lower convergence rate. Thus, in this work we introduce a method that compromises between the recovery error and convergence speed by using PGD with an inexact "projection" that projects onto a set that is approximately as small as $\hat{\mathcal{K}}$ but yet is as computationally efficient as the projection onto $\mathcal{K}$. In this way, the computational complexity

Authorized licensed use limited to: Institute of Science and Technology Austria (IST). Downloaded on September 22,2026 at 10:55:56 UTC from IEEE Xplore. Restrictions apply.

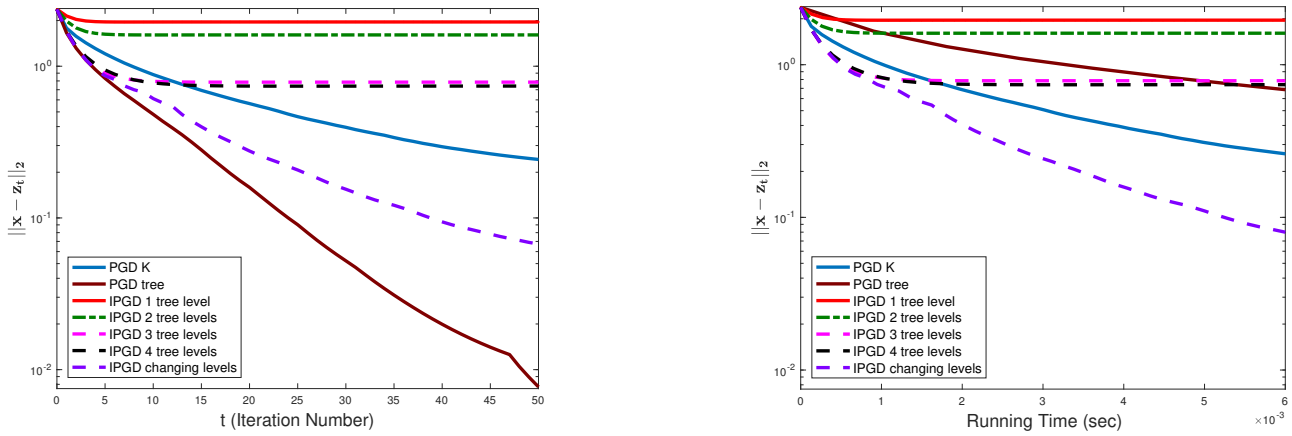


Fig. 1. Reconstruction error as a function of the iterations (left) and the running time (right) for recovering a sparse vector with tree structure. As we initialize all algorithms with the zero vector, the error at iteration/time zero is $\|\mathbf{x}\|$.

of each PGD iteration remains the same but the convergence rate becomes closer to the one with a projection onto $\hat{\mathcal{K}}$.

Our "projection" is composed of a simple operator p (e.g., linear) and the projection onto $\mathcal{K}$, $\mathcal{P}_{\mathcal{K}}$, such that it introduces only a slight distortion into $\mathbf{x}$. If $\mathcal{K}$ is convex, then we require

$$\|\mathbf{x} - \mathcal{P}_{\mathcal{K}}\left(p(\mathbf{x})\right)\| \leq \epsilon \|\mathbf{x}\| . \tag{14}$$

and if $\mathcal{K}$ is non-convex, we require

$$\|\mathcal{P}_{\mathcal{K}}\left(p\mathbf{v} - p\mathbf{x}\right) - \mathcal{P}_{\mathcal{K}}\left(p\mathbf{v} - \mathbf{x}\right)\| \leq \epsilon \|\mathbf{x}\| , \forall \mathbf{v} \in \mathbb{R}^d. \tag{15}$$

An example for p that satisfies (15) appears in Section 5. Plugging the inexact projection into the PGD step results in the proposed IPGD iteration (compare to (7)),

$$\mathbf{z}_{t+1} = \mathcal{P}_{\mathcal{K}}\left(p\left(\mathbf{z}_t\right) + \mu p\left(\mathbf{M}^*(\mathbf{y} - \mathbf{M}\mathbf{z}_t)\right)\right). \tag{16}$$

4. IPGD CONVERGENCE ANALYSIS

We now move to analyze the performance of IPGD. For simplicity of the discussion, we analyze the convergence of this technique only for a linear operator p and the noiseless setting, i.e., $\mathbf{e} = 0$. The extension to other types of operators and the noisy case is straightforward by arguments similar to those used in [22] for treating the noise term and other classes of matrices. The proof of the theorem appears in [18].

Theorem 4.1 *Let $\mathbf{x} \in \mathcal{K}$, $\mathcal{K} \subset \mathbb{R}^d$ be a closed cone, $p(\cdot)$ a linear operator satisfying* (15)*, $\mathbf{M} \in \mathbb{R}^{m \times d}$ and $\mathbf{y} = \mathbf{M}\mathbf{x}$ a vector containing m linear measurements. Assume we are using IPGD with $\mathcal{K}$ and p to recover $\mathbf{x}$ from $\mathbf{y}$. Then the estimate $\mathbf{z}_t$ at the tth iteration (initialized with $\mathbf{z}_0 = 0$) obeys*

$$\|\mathbf{z}_t - \mathbf{x}\| \leq \left(\left(\kappa_{\mathcal{K}} \rho_p(\mathcal{K})\right)^t + \frac{1 - \left(\kappa_{\mathcal{K}} \rho_p(\mathcal{K})\right)^t}{1 - \kappa_{\mathcal{K}} \rho_p(\mathcal{K})} \gamma \right) \|\mathbf{x}\| , \tag{17}$$

where $\kappa_{\mathcal{K}}$ and $\rho(K)$ are defined in Theorem 2.1, $\gamma \triangleq (2\rho(\mathcal{K})\kappa_{\mathcal{K}} + \rho_p(\mathcal{K})\kappa_{\mathcal{K}} + 1)\epsilon$, and

$$\rho_p(\mathcal{K}) = \sup_{\mathbf{u},\mathbf{v} \in (\mathcal{K}-\mathcal{K}) \cap \mathbb{B}^d} p(\mathbf{u})^* \left(\mathbf{I} - \mu \mathbf{M}^* \mathbf{M}\right) p(\mathbf{v}) \tag{18}$$

sets the "effective convergence rate" of the IPGD for small ϵ.

Theorem 4.1 implies that if ϵ is small enough compared to ρ_p^t, then IPGD has an effective convergence rate of ρ_p. Note that if $p = \mathbf{I}$ then $\epsilon = 0$ and we get back to Theorem 2.1. As we shall see hereafter, for some operators p the rate ρ_p may be significantly smaller than $\rho(\mathcal{K})$. The smaller the set that p maps to, the smaller ρ_p becomes. Yet, the approximation error ϵ in (15) increases. Thus, IPGD allows us to tradeoff approximation error ϵ and improved convergence ρ_p.

The error term in Theorem 4.1 comprises of two components. The first goes to zero as t increases while the second increases with iterations and is of the order of ϵ. The fewer iterations we perform the larger ϵ we may allow. Looking at it from another perspective, the larger the reconstruction error we can tolerate the larger ϵ can be and thus the fewer iterations we need. Therefore, the projection p introduces a tradeoff. On the one hand, it brings additional error to the reconstruction error. On the other hand, it simplifies the projected set, which leads to a faster convergence (to the larger error).

5. EXPERIMENTS

Sparse recovery with tree structure. To demonstrate our theory we consider a variant of the k-sparse set with tree structure in (13) that has smaller weights in the lower nodes of the tree. We generate a k-sparse vector $\mathbf{x} \in \mathbb{R}^{127}$ with $k = 13$ and a sparsity pattern that obeys a tree structure. We set the non-zero entries in $\mathbf{x}$ independently from a Gaussian distribution with zero mean and variance $\sigma^2 = 1$ if they are at the first two levels of the tree and $\sigma^2 = 0.2^2$ for the rest.

Authorized licensed use limited to: Institute of Science and Technology Austria (IST). Downloaded on September 22,2026 at 10:55:56 UTC from IEEE Xplore. Restrictions apply.

The best way to recover $\mathbf{x}$ is by using a projection onto the set $\hat{\mathcal{K}}$ in (13), which is used in model-based compressed sensing [21]. Yet, this projection requires some additional computations at each iteration. Our method suggests to approximate it by a simple linear projection p onto the first levels of the tree followed by a projection onto $\mathcal{K} = \{\mathbf{z} : \|\mathbf{z}\|_0 \leq k\}$. It is easy to show that $\epsilon \leq 2\frac{\|p(\mathbf{x})-\mathbf{x}\|}{\|\mathbf{x}\|}$ in this case (see (15)).

Clearly, the more levels we add in the projection p, the smaller the approximation error ϵ becomes. Yet, assuming that all nodes in each layer are selected with equal probability, then the probability of selecting a node at layer l is equal to $\prod_{i=1}^{l} 0.5^{i-1}$, where we take here into account the fact that a node can be selected only if all its forefathers have been selected. Thus, the upper layers have more significant impact on the values of ϵ. On the other hand, the convergence rate $\rho_p(\mathcal{K})$ for a projection with l layers is equivalent to the convergence rate for the set of vectors of size 2^l (denoted by $\mathcal{K}_l$). Thus, we get that $\rho_p(\mathcal{K}) = \rho(\mathcal{K}_l)$, which is dependent on the Gaussian mean width $\omega_{\mathcal{K}_l}$ that scales as $\max(kl, k\log(2^l/k))$. Clearly, when we take all the layers $l = \log(d)$ and we have $\omega_{\mathcal{K}_l} = \omega_{\mathcal{K}} = O(k\log(d/k))$.

Figure 1 presents the signal reconstruction error ($\|\mathbf{x} - \mathbf{z}_t\|_2$) as a function of the number of iterations for PGD with the sets $\mathcal{K}$ (IHT [20]) and $\hat{\mathcal{K}}$ (model-based IHT [21])[1] and for the proposed IPGD with p that projects onto a different number of levels (1-5) of the tree. All algorithms use step size $\mu = \frac{1}{(\sqrt{d}+\sqrt{m})^2}$. It is interesting to note that if p projects only onto the first layer, then the algorithm does not converge as the resulting approximation error ϵ is too large. However, starting from the second layer, we get a faster convergence at the first iterations with p that projects onto a smaller set, which yields a smaller ρ. Yet, as the number of iterations increases, the more accurate projections attain a lower reconstruction error, where the plateau attained is proportional to the approximation error of p as predicted by our theory.

This tradeoff can be used to further accelerate the convergence by changing the projection in IPGD over the iterations. Thus, in the first iterations we enjoy the fast convergence of the coarser projections and in the later ones we use more accurate projections that allow achieving a lower plateau. The last line in Fig. 1 demonstrates this strategy, where at the first iteration p is set to be a projection onto the first two levels, and then every four iterations another tree level is added to the projection until it becomes a projection onto all the levels (in this case IPGD coincides with PGD). Notice that while this method converges slower than PGD with $\hat{\mathcal{K}}$ with respect to the iteration number, it is faster in its running time.

5.1. Learning the projection – Learned IPGD (LIPGD)

In many scenarios, we may not know what type of simple operator p causes $\mathcal{P}_{\mathcal{K}}(p(\cdot))$ to approximate $\hat{\mathcal{K}}$ in the best possible way. Therefore, a useful strategy is to learn p for a given dataset. Assuming a linear p, we may rewrite (16) as

$$\mathbf{z}_{t+1} = \mathcal{P}_{\mathcal{K}}\left(p\left(\mu\mathbf{M}^*\mathbf{y}\right) + p\left(\left(\mathbf{I} - \mu\mathbf{M}^*\mathbf{M}\right)\mathbf{z}_t\right)\right). \quad (19)$$

Instead of learning p directly, we may learn two matrices $\mathbf{A}$ and $\mathbf{U}$ that replace $p\mu\mathbf{M}^*$ and $p\left(\mathbf{I} - \mu\mathbf{M}^*\mathbf{M}\right)$ respectively. This results in the LIPGD iteration

$$\mathbf{z}_{t+1} = \mathcal{P}_{\mathcal{K}}\left(\mathbf{A}\mathbf{y} + \mathbf{U}\mathbf{z}_t\right), \quad (20)$$

which is very similar to LISTA (see (6)). The only difference between (20) and (6) is the non-linearity: It is an orthogonal projection in the first and a proximal mapping in the second.

We use this method to replace the sparse coding step in the super-resolution algorithm proposed in [17], where a pair of low-res and high-res dictionaries are used for reconstructing the patches of the high resolution image from the low-resolution one. In the code provided by the authors, OMP with sparsity 3 is used. The complexity of this strategy corresponds to IHT with 3 iterations, which we apply with higher target sparsity level that is observed to provide better reconstruction results. Note that in IHT, unlike OMP, the number of iterations may be different than the sparsity level. As IHT does not convergence with only 3 iterations, we use LIPGD to get accelerated convergence. We train it using 91 images from the dataset used for training the dictionary in [17] being split into a training set of size 86 and a validation set of size 5 (used for tuning the hyper-parameters of the LIPGD such as the target sparsity). Table 1 presents the advantage of using LIPGD over generic bicubic interpolation and IHT and OMP (with 3 iterations) for sparse coding.

Image	Bicubic	OMP	IHT	LIPGD
baboon	23.2	23.5	23.4	**23.6**
bridge	24.4	25.0	24.8	**25.1**
coastguard	26.6	27.1	26.9	**27.2**
comic	23.1	24.0	23.8	**24.2**
face	32.8	33.5	33.2	**33.6**
flowers	27.2	28.4	28.1	**28.7**
foreman	31.2	33.2	32.3	**33.5**
lenna	31.7	33.0	32.6	**33.2**
man	27.0	27.9	27.7	**28.1**
monarch	29.4	31.1	30.9	**31.6**
pepper	32.4	34.0	33.6	**34.4**
ppt3	23.7	25.2	24.6	**25.5**
zebra	26.6	28.5	28.0	**28.9**

Table 1. PSNR of super-resolution by bicubic interpolation and a pair of dictionaries with various sparse coding methods.

6. CONCLUSION

In this work we developed a theory that offers an explanation to the recent success of neural networks for approximating the solution of certain minimization problems. We demonstrate the usage of our result in the context of sparse recovery with tree structure and in the problem of image super resolution.

[1] for demonstration purposes we plot only the cases where model-based IHT converges to zero.

Authorized licensed use limited to: Institute of Science and Technology Austria (IST). Downloaded on September 22,2026 at 10:55:56 UTC from IEEE Xplore. Restrictions apply.

7. REFERENCES

[1] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM Review*, vol. 51, no. 1, pp. 34–81, 2009.

[2] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, H. H. Bauschke, S. R. Burachik, L. P. Combettes, V. Elser, R. D. Luke, and H. Wolkowicz, Eds. 2011, pp. 185–212, Springer New York.

[3] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *arXiv:1606.04838*, 2016.

[4] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Img. Sci.*, vol. 2, no. 1, pp. 183–202, Mar. 2009.

[5] M. Elad, B. Matalon, and M. Zibulevsky, "Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization," *Appl. Comput. Harmon. Anal.*, vol. 23, no. 3, pp. 346 – 367, 2007.

[6] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004.

[7] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $o(1/k^2)$," *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372 –376, 1983.

[8] E. Treister and I. Yavneh, "A multilevel iterated-shrinkage approach to ℓ_1 penalized least-squares minimization," *IEEE Trans. Signal Process.*, vol. 60, no. 12, pp. 6319–6329, Dec. 2012.

[9] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2010.

[10] P. Sprechmann, A. M. Bronstein, and G. Sapiro, "Learning efficient sparse and low rank models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1821–1833, Sept. 2015.

[11] T. Remez, O. Litany, and A. M. Bronstein, "A picture is worth a billion bits: Real-time image reconstruction from dense binary pixels," in *18th International Conference on Computational Photography (ICCP)*, 2015.

[12] J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin, "Accelerating Eulerian fluid simulation with convolutional networks," *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

[13] M. Andrychowicz, M.and Denil, S. Gómez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas, "Learning to learn by gradient descent by gradient descent," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 3981–3989. 2016.

[14] T. Moreau and J. Bruna, "Understanding neural sparse coding with matrix factorization," *ICLR*, 2017.

[15] E.J. Candès and T. Tao, "Decoding by linear programming," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4203 – 4215, dec. 2005.

[16] M. Borgerding, P. Schniter, and S. Rangan, "Amp-inspired deep networks for sparse linear inverse problems," *IEEE Transactions on Signal Processing*, vol. 65, no. 16, pp. 4293–4308, Aug 2017.

[17] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Proceedings of the 7th international conference on Curves and Surfaces*, Berlin, Heidelberg, 2012, pp. 711–730, Springer-Verlag.

[18] R. Giryes, Y. C. Eldar, A. M. Bronstein, and G. Sapiro, "Tradeoffs between convergence speed and reconstruction accuracy in inverse problems," *to appear in IEEE Trans. Sig. Proc.*, 2018.

[19] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient projections onto the ℓ_1-ball for learning in high dimensions," in *Proceedings of the 25th Annual International Conference on Machine Learning (ICML)*, 2008.

[20] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Appl. Comput. Harmon. Anal*, vol. 27, no. 3, pp. 265 – 274, 2009.

[21] R.G. Baraniuk, V. Cevher, M.F. Duarte, and C. Hegde, "Model-based compressive sensing," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1982–2001, April 2010.

[22] S. Oymak, B. Recht, and M. Soltanolkotabi, "Sharp time–data tradeoffs for linear inverse problems," *to appear in IEEE Trans. Inf. Theory*, 2018.

[23] T. Tirer and R. Giryes, "Generalizing cosamp to signals from a union of low dimensional linear subspaces," *arXiv abs/1703.01920*, 2017.

[24] Y. Plan and R. Vershynin, "Robust 1-bit compressed sensing and sparse logistic regression: A convex programming approach," *IEEE Trans. Inf. Theory*, vol. 59, no. 1, pp. 482–494, Jan. 2013.

Authorized licensed use limited to: Institute of Science and Technology Austria (IST). Downloaded on September 22,2026 at 10:55:56 UTC from IEEE Xplore. Restrictions apply.