Architectural Adversarial Robustness: The Case for Deep Pursuit

George Cazenavette[1] Calvin Murdock[1] Simon Lucey[1,2]
[1]Carnegie Mellon University [2]University of Adelaide
{gcazenav,cmurdock,slucey}@cs.cmu.edu

Abstract

Despite their unmatched performance, deep neural networks remain susceptible to targeted attacks by nearly imperceptible levels of adversarial noise. While the underlying cause of this sensitivity is not well understood, theoretical analyses can be simplified by reframing each layer of a feed-forward network as an approximate solution to a sparse coding problem. Iterative solutions using basis pursuit are theoretically more stable and have improved adversarial robustness. However, cascading layer-wise pursuit implementations suffer from error accumulation in deeper networks. In contrast, our new method of deep pursuit approximates the activations of all layers as a single global optimization problem, allowing us to consider deeper, real-world architectures with skip connections such as residual networks. Experimentally, our approach demonstrates improved robustness to adversarial noise.

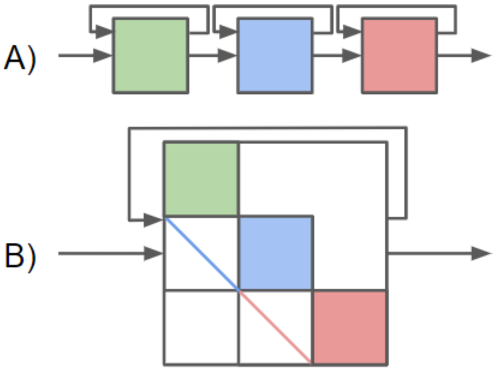

Figure 1. (a) Recurrent deep networks that implement layered basis pursuit have been shown to be provably more robust to adversarial noise than feed-forward alternatives. However, this method is incompatible with modern architectures. (b) We instead propose deep pursuit, which jointly infers all network activations as a single structured sparse coding problem.

1. Introduction

Multilayer sparse approximation has been proposed as a robust alternative to feed-forward neural networks [17]. While provably less sensitive to noise, recurrent networks that implement layered basis pursuit accumulate independent errors and cannot be applied to modern large-scale architectures. We propose a new method of *deep pursuit*, wherein all activations of the network are synchronously optimized through a global basis pursuit, circumventing error accumulation and accounting for the skip connections commonly found in state-of-the-art network architectures.

We apply this technique to address a major weakness of deep neural networks: despite unrivaled performance on supervised tasks, they can be highly sensitive to certain types of data noise. Specifically, adversarial attacks use imperceptible targeted input perturbations to completely change a network's predictions [8]. Robustness to such attacks is mission-critical to many domains, such as security systems and autonomous vehicles.

Because the generalization properties of deep neural networks are not yet thoroughly understood, combating such adversarial attacks remains an open problem. Current state-of-the-art methods rely on specialized loss functions or training techniques. However, these methods do not explain *why* some models are more susceptible to attacks than others or how to create naturally robust architectures. In this work, we apply techniques from sparse approximation theory to design deep neural networks that are intrinsically more robust to adversarial noise.

Previous works have suggested reframing each layer of a neural network as a sparse coding problem to make their outputs more robust [17] (Figure 1a). However, as noted by the authors, this method accumulates error throughout the layers of the network, potentially leading to poor performance in deeper models. Additionally, this method offers no provisions for handling skip connections between layers, preventing its use for real-world network architectures.

In order to exploit the natural robustness offered by deeper networks [19] and skip connections [9], we propose adapting the layer-wise pursuit algorithm introduced in [17] to the global view that reframes a neural network as an approximate solution to a single sparse coding problem with block-structured parameters (Figure 1 B) [14]. We optimize the outputs of all layers synchronously [4], which effectively amounts to adding recurrent feedback connections on top of a feed-forward network. By relating the entire network to a single structured sparse coding problem, our method does not suffer from error accumulation as the network grows deeper. Furthermore, we can even entertain residual and dense skip connections within our optimization, something not possible using layered basis pursuit. We call this method "deep pursuit."

Our contributions are:

1. Through connections to sparse approximation theory, we illustrate how the structure of a global sparse approximation problem predicts why certain architectures are naturally more robust.
2. Extending the method of layered basis pursuit to our global view, we propose a technique for synchronously inferring all latent activations via block coordinate descent.
3. We show how deep pursuit outperforms layered basis pursuit by avoiding error accumulation and allowing for skip connections between layers. Experimentally, we demonstrate improved robustness to adversarial attacks on the CIFAR-10 dataset.

2. Related Works

2.1. Adversarial Examples

In a white-box setting where an attacker has access to the model's parameters, an adversarial example can be crafted to modify a prediction by explicitly maximizing the model's loss within given bounds on the noise. Goodfellow et. al. introduced the "fast gradient sign method" where additive perturbations are constructed from the *sign* of the gradient of the output with respect to the input [8]. In a purely linear model, this maximizes the change in the output. They hypothesized that this attack translates so well to neural networks because their components are all quasi-linear, despite the overall function being technically highly non-linear.

2.2. Adversarial Training

Current state of the art methods for training models robust to adversarial attacks like these work by creating loss-maximizing adversarial examples at train time [7, 18, 22]. By doing so, the model learns to correctly classify or embed such examples. However, one caveat to this method is that it only *directly* encourages robustness towards the type of attack used to generate the adversarial samples.

Additionally, adversarial training takes much longer to converge. Since the training set is continuously being updated along with the model, the objective function is non-stationary. This results in the optimization chasing an ever-moving target, raising questions of when it is an appropriate time to stop training. Evaluating adversarial training methods also necessitates testing performance on "seen" versus "unseen" attacks. Since our proposed deep pursuit method is purely architectural, we circumvent this requirement.

2.3. Sparse Approximation

Sparse coding techniques are useful in signal representation tasks in that they are *provably robust*. In contrast to feed-forward representations that may amplify input errors, iterative optimization of a sparsity-inducing objective function can be provably insensitive to input noise [6]. Building upon theoretical connections between feed-forward deep networks and sparse coding [15], recent work has even shown that using a supervised sparse encoder for classification tasks theoretically bounds the adversarial error [20].

The robustness of sparse coding techniques relies on the redundancy of overcomplete representations. Their effectiveness is determined by the mutual coherence of the reconstruction dictionaries, or the maximum absolute normalized inner product of the atoms used in sparse linear combinations for approximating input data [17, 20]. Small mutual coherence leads to dictionaries that are closer to orthogonal. Murdock and Lucey [14] developed a method of analyzing the global mutual coherence of deep neural networks by viewing the parameters of all layers as a single structured matrix. Adding more layers with denser connections between them decreases an architecture-dependent lower bound on the global mutual coherence. Through correlations with generalization capacity, this provided an explanation for why deeper networks and those with skip connections are more naturally robust, leading to improved generalization performance without overfitting.

2.4. Low-Rank Representations

Another method of achieving robustness is by exploiting low-rank embeddings of the data [1]. It has even been shown that dropout, an implicit method of encouraging robustness with noise, is related to low-rank weight matrix factorization [3].

3. Architectural Robustness

Shallow iterative sparse approximation with layered basis pursuit has been shown to be provably more robust than a feed-forward neural network layer [17]. By adapting this theory towards a global view of a deep neural network as a single structured sparse coding problem, we aim to construct neural networks that are even more robust.

3.1. Neural Networks as Sparse Coding

To view a neural network as an approximate solution to a sparse coding problem, we must first see the link between proximal operators and non-linear activation functions. Be-

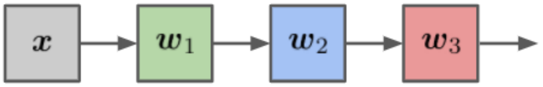

Figure 2. A feed-forward neural network can be reframed as a cascade of sparse coding problems with solutions approximated via layered thresholding pursuit.

cause its trust region encourages sparsity in the trained parameters, the ℓ_1 regularizer is often used as a surrogate for minimizing the (intractable) ℓ_0 norm [5]. The proximal operator, a generalization of projection onto constraints, is a tool used in convex optimization to optimize objectives with non-differentiable penalty functions [16]. The proximal operator of the ℓ_1 norm with weight $\lambda > 0$ yields the elementwise soft thresholding operator:

$$\phi_\lambda(\boldsymbol{x}) = \begin{cases} \boldsymbol{x} - \lambda & \boldsymbol{x} > \lambda \\ 0 & -\lambda \leq \boldsymbol{x} \leq \lambda \\ \boldsymbol{x} + \lambda & \boldsymbol{x} < -\lambda \end{cases} \tag{1}$$

Papayan et. al. [15] showed that if we also apply a non-negative constraint, the resulting proximal operator $\tilde{\phi}_\lambda$ in Eq. 2 is equivalent to the Rectified Linear Unit (ReLU), a nonlinearity commonly used in many state-of-the art deep networks, with a negative bias of λ.

$$\tilde{\phi}_\lambda(\boldsymbol{x}) = \arg\min_{\boldsymbol{w}\geq 0} \tfrac{1}{2}\|\boldsymbol{x}-\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_1 = \mathrm{ReLU}(\boldsymbol{x}-\lambda) \tag{2}$$

As such, we can then reframe a single layer of a neural network as the approximate solution of the following non-negative sparse coding problem (non-negative LASSO [21]) solved via soft-thresholding pursuit:

$$\min_{\boldsymbol{w}\geq 0} \tfrac{1}{2}\|\boldsymbol{x} - \mathbf{B}\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_1 \tag{3}$$

A feed-forward chain network (without skip connections) with ReLU activations can then be interpreted as layered soft-thresholding pursuit, a cascade of approximate solutions to sparse-coding problems (Figure 2):

$$f(\boldsymbol{x}) = \tilde{\phi}_{\lambda_l}(\mathbf{B}_l^T \ldots \tilde{\phi}_{\lambda_2}(\mathbf{B}_2^T \tilde{\phi}_{\lambda_1}(\mathbf{B}_1^T \boldsymbol{x}))\ldots) \tag{4}$$

Here, $\boldsymbol{x}$ is an input vector and $\mathbf{B}_i$ are (dense or convolutional) parameter dictionaries.

In this setting, the parameters are learned as

$$\arg\min_{\mathbf{A},\{\mathbf{B}_j\},\{\lambda_j\}} \sum_{i=1}^{n} J(\mathbf{A}^T f(\boldsymbol{x}_i), y_i) \tag{5}$$

where J is a loss function (e.g. cross-entropy), $\mathbf{A}$ contains the parameters of a linear classifier, and y_i is the target for sample $\boldsymbol{x}_i$. One drawback of this classic feed-forward design is that small perturbations in the input $\boldsymbol{x}$ can be amplified through the layers leading to large changes in the output $f(\boldsymbol{x})$ for increased sensitivity to adversarial noise.

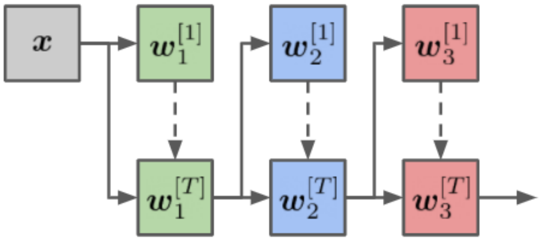

Figure 3. Instead of thresholding pursuit, Romano et. al. [17] proposed solving each layer's sparse coding problem with an iterative algorithm. However, this method of layered basis pursuit suffers from error accumulation and cannot account for skip connections.

By expressing neural networks as cascades of approximate sparse coding problems, we can create more robust models by reducing the noise sensitivity of each individual layer in the network.

3.2. Local Iterations

One of the most popular algorithms used to solve the LASSO problem is the Iterative Shrinking and Thresholding Algorithm (ISTA) [2]. ISTA's iterative update takes the following form by first taking a negative (reconstruction loss) gradient step then applying the proximal operator defined in Eq. 1:

$$\boldsymbol{w}^{[t+1]} = \phi_\lambda(\boldsymbol{w}^{[t]} - \mathbf{B}^T(\mathbf{B}\boldsymbol{w}^{[t]} - \boldsymbol{x})) \tag{6}$$

When considering non-negative ISTA, we simply replace the soft thresholding operator ϕ_λ with the non-negative soft thresholding operator $\tilde{\phi}_\lambda$.

Adopting a sparse-coding view of deep learning, Romano et. al. [17] re-framed a deep neural network as a cascade of sparse approximation problems solved with an iterative basis pursuit algorithm. Specifically, for an l-layer neural network f such that $f(\boldsymbol{x}) = \boldsymbol{w}_l$, for each layer j, we have that

$$\boldsymbol{w}_j := \arg\min_{\boldsymbol{w}\geq 0} \tfrac{1}{2}\|\boldsymbol{w}_{j-1} - \mathbf{B}_j\boldsymbol{w}\|_2^2 + \lambda_j\|\boldsymbol{w}\|_1 \tag{7}$$

where $\boldsymbol{w}_{j-1}$ is the output of the previous layer, $\boldsymbol{w}_0 = \boldsymbol{x}$, and $\boldsymbol{w} \geq 0$ constrains all values of $\boldsymbol{w}$ to be non-negative. Semantically, $\boldsymbol{w}$ are coefficients used to reconstruct the coefficients of the previous layer, just as ISTA optimizes coefficients to reconstruct a signal given a dictionary.

To solve this layer-wise optimization problem, we implement proximal gradient descent where the gradient of the smooth reconstruction loss, $\boldsymbol{g}_j^{[t]}$, is

$$\boldsymbol{g}_j^{[t]} = \frac{\partial}{\partial \boldsymbol{w}_j}\|\boldsymbol{w}_{j-1} - \mathbf{B}_j\boldsymbol{w}_j\|_2^2 = \mathbf{B}_j^T(\mathbf{B}_j\boldsymbol{w}_j^{[t-1]} - \boldsymbol{w}_{j-1}) \tag{8}$$

and the algorithm is initialized with the feed-forward soft-thresholding pursuit approximation

$$\boldsymbol{w}_j^{[0]} = \tilde{\phi}_{\lambda_j}(\mathbf{B}_j^T\boldsymbol{w}_{j-1}^{[0]}) = \mathrm{ReLU}(\mathbf{B}_j^T\boldsymbol{w}_{j-1}^{[0]} - \lambda_j) \tag{9}$$

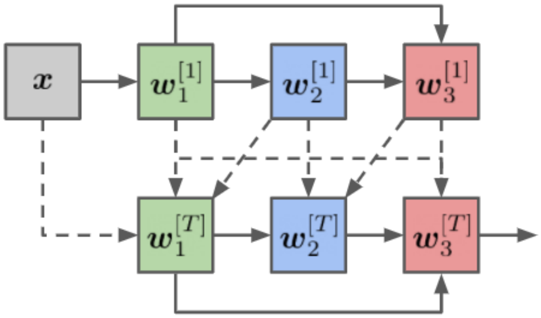

Figure 4. In deep pursuit, all layers are updated synchronously and take feedback from all adjacent layers. This method eliminates error accumulation and accounts for skip connections (as seen between layers 1 and 3).

which is iteratively updated by step size γ_j as

$$\boldsymbol{w}_j^{[t]} = \tilde{\phi}_{\lambda_j}(\boldsymbol{w}_j^{[t-1]} - \gamma_j \mathbf{B}_j^T(\mathbf{B}_j \boldsymbol{w}_j^{[t-1]} - \boldsymbol{w}_{j-1})) \tag{10}$$

Here, γ_j is initialized to $\frac{1}{L_j}$ where L_j is the conservative Lipschitz bound for guaranteed convergence as derived in [4]. However, since we do not iterate until convergence, we introduce a trainable parameter $\beta_j \in (0, 1]$ that allows the network to automatically learn a larger step size such that $\gamma_j = \frac{1}{\beta_j L_j}$. Our implementation of the full algorithm is shown in Algorithm 1. In practice, it can be implemented as a recurrent network where each layer is unrolled to a fixed number of iterations (Figure 3.)

It was hypothesized in [17] that explicitly solving a sparse coding problem makes the representation at each layer more stable, and our empirical results corroborate this theory. However, as noted in [17], any error remaining is compounded through the subsequent layers of the network, making this method less effective for sufficiently deep networks. Furthermore, it is not clear how local iterations (layered basis pursuit) could be adapted to network architectures with skip connections, making this method infeasible for modern architectures.

Algorithm 1: Inference with layered thresholding pursuit [17]. See Eq. 8 for a definition of $\boldsymbol{g}$.

```
$\boldsymbol{w}_0 \leftarrow \boldsymbol{x}$
for $j \leftarrow 1$ to $l$ do
    for $t \leftarrow 1$ to $T$ do
        $\boldsymbol{w}_j^{[t]} \leftarrow \tilde{\phi}_{\lambda_j}(\boldsymbol{w}_j^{[t-1]} - \frac{1}{L_j \beta_j} \boldsymbol{g}_j^{[t]})$
    end
end
```

3.3. Deep Pursuit

As shown in Eq. 4, a neural network can be expressed as a cascade of multiple sparse coding problem. Alternatively, we can view the activations of an entire network as an approximate solution to a single sparse coding problem. In this case, we can take our global loss as the sum of each layer's loss and infer all coefficients synchronously.

In this case, the global multi-layer non-negative LASSO objective is:

$$\underset{\{\boldsymbol{w}_j \geq 0\}}{\arg\min} \frac{1}{2} \sum_{j=1}^{l} \|\boldsymbol{w}_{j-1} - \mathbf{B}_j \boldsymbol{w}_j\|_2^2 + \lambda_j \|\boldsymbol{w}_j\|_1 \tag{11}$$

Viewed with a single structured parameter matrix [14], this can be equivalently expressed as

$$\underset{\{\boldsymbol{w}_j \geq 0\}}{\arg\min} \frac{1}{2} \left\| \begin{bmatrix} \boldsymbol{x} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{B}_1 & & & \mathbf{0} \\ -\mathbf{I} & \mathbf{B}_2 & & \\ & \ddots & \ddots & \\ \mathbf{0} & & -\mathbf{I} & \mathbf{B}_l \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \\ \vdots \\ \boldsymbol{w}_l \end{bmatrix} \right\|_2^2 + \sum_{j=1}^{l} \lambda_j \|\boldsymbol{w}_j\| \tag{12}$$

Now, instead of relying on a cascading composition of solutions, we can infer all coefficients jointly by solving a single shallow sparse coding problem.

Structure-agnostic algorithms for shallow solutions (like soft thresholding pursuit) will not work in this case since there is no way for information to propagate from the input $\boldsymbol{x}$ to the output $f(\boldsymbol{x})$. Instead, we can now group the variables and apply the block coordinate descent algorithm introduced in [23] and applied in [4]. This optimization is similar to that of local iterations, except that it incorporates feedback from connected layers, and every layer is updated in sequence during each iteration.

In this setting, the update for each layer j during global iteration t is given as

$$\boldsymbol{w}_j^{[t]} = \tilde{\phi}_{\lambda_j}(\hat{\boldsymbol{w}}_j^{[t-1]} - \frac{1}{\beta_j L_j} \hat{\boldsymbol{g}}_{j-1}^{[t]}) \tag{13}$$

where $\hat{\boldsymbol{w}}$ is the previous iteration's result extrapolated with $0 \leq \alpha_j < 1$ as in [4] for improved convergence speed:

$$\hat{\boldsymbol{w}}_j^{[t-1]} = \boldsymbol{w}_j^{[t-1]} + \alpha_j(\boldsymbol{w}_j^{[t-1]} - \boldsymbol{w}_j^{[t-2]}) \tag{14}$$

The gradient of the global reconstruction error with respect to block j is defined as

$$\hat{\boldsymbol{g}}_j^{[t]} = \begin{cases} \mathbf{B}_j^T(\mathbf{B}_j \hat{\boldsymbol{w}}_j^{[t-1]} - \boldsymbol{w}_{j-1}^{[t]}) & \\ \quad +(\hat{\boldsymbol{w}}_j^{[t-1]} - \mathbf{B}_{j+1} \boldsymbol{w}_{j+1}^{[t-1]}) & j < l \\ \mathbf{B}_j^T(\mathbf{B}_j \hat{\boldsymbol{w}}_j^{[t-1]} - \boldsymbol{w}_{j-1}^{[t]}) & j = l \end{cases} \tag{15}$$

We see that the gradient for the global update shown in Eq. 15 is similar for the gradient in the layer-wise update shown in Eq. 10. The key distinction is that the global update takes into account feedback from the subsequent layer's result from the previous iteration because there are now two terms that include $\boldsymbol{w}_j$ since the following layer is trying to reconstruct it.

Since all parameters are updated synchronously, we no longer have the problem of error accumulation present in layered basis pursuit [17]. This allows us to entertain deeper networks without fear of exploding error in cascading solutions. While the robustness of layered basis pursuit is determined by the mutual coherence of each layer [17], the robustness of global deep pursuit is instead determined by the mutual coherence of the global structured dictionary.

We give an outline of our deep pursuit algorithm in Algorithm 2. Note that the inner and outer loops are reversed so that the output of each layer is now updated at each iteration. In contrast to the layered pursuit algorithm in Figure 3, observe that the deep pursuit algorithm in Figure 4 allows for skip connections. The feedback connections between layers alleviate the information bottleneck between layers, facilitating more effective backpropogation in deeper networks and inducing a less coherent global dictionary.

Algorithm 2: Inference with deep pursuit. See Eqs. 15 and 18 for definitions of $\hat{\boldsymbol{g}}$.

```
$\boldsymbol{w}_0 \leftarrow \boldsymbol{x}$
for $j \leftarrow 1$ to $l$ do
    $\boldsymbol{w}_j^{[0]} \leftarrow \tilde{\phi}_{\lambda_j}(\mathbf{B}_j^T \boldsymbol{w}_{j-1}^{[0]})$
end
for $t \leftarrow 1$ to $T$ do
    for $j \leftarrow 1$ to $l$ do
        $\boldsymbol{w}_j^{[t]} \leftarrow \tilde{\phi}_{\lambda_j}(\hat{\boldsymbol{w}}_j^{[t-1]} - \frac{1}{\beta_j L_j} \hat{\boldsymbol{g}}_{j-1}^{[t]})$
    end
end
```

3.4. Global Iterations with Skip Connections

Skip connections between layers have become key components of nearly all state-of-the-art architectures [10, 11]. From the perspective of sparse approximation, denser skip connections between layers induce global dictionary structures with lower mutual coherence [14], leading to improved robustness even with feed-forward approximations. While the method of layered basis pursuit [17] gives no clear way to incorporate these skip connections, deep pursuit can be naturally adapted to support skip connections between layers.

To accomplish this, we can modify the global structured dictionary matrix in Eq. 12 to account for general skip connections by including additional off-diagonal blocks of parameters $\mathbf{B}_{jk}$ connecting layers j and k:

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 & & & \mathbf{0} \\ -\mathbf{B}_{21}^T & \mathbf{B}_2 & & \\ \vdots & \ddots & \ddots & \\ -\mathbf{B}_{l1}^T & \cdots & -\mathbf{B}_{l(l-1)}^T & \mathbf{B}_l \end{bmatrix} \tag{16}$$

Taking advantage of the block lower-diagonal structure, approximate feed-forward inference for sparse coding problems with dictionaries like these can proceed incrementally through the network [14]. Specifically, the output $\boldsymbol{w}_j$ of layer j can be found given all previous outputs as:

$$\begin{aligned} \boldsymbol{w}_j &= \tilde{\phi}_{\lambda_j}\Big(\mathbf{B}_j^T \sum_{k=1}^{j-1} \mathbf{B}_{jk}^T \boldsymbol{w}_k\Big) \\ &\approx \operatorname*{arg\,min}_{\boldsymbol{w}_j} \Big\| \mathbf{B}_j \boldsymbol{w}_j - \sum_{k=1}^{j-1} \mathbf{B}_{jk}^T \boldsymbol{w}_k \Big\|_2^2 + \lambda_j \|\boldsymbol{w}_j\|_1 \end{aligned} \tag{17}$$

This equation gives the general case for dense connections between every layer. For example, a residual connection between non-adjacent layers j and k is represented by $\mathbf{B}_{jk} = \mathbf{I}$ since the coefficients of layer k are simply added to the pre-activations of layer j. Further details on these structured dictionary matrices can be found in [14].

With this denser global dictionary, our gradients now include feedback using the most recent updates from all connected layers (Figure 4):

$$\begin{aligned} \hat{\boldsymbol{g}}_j &= \mathbf{B}_j^T(\mathbf{B}_j \boldsymbol{w}_j - \sum_{k=1}^{j-1} \mathbf{B}_{jk}^T \boldsymbol{w}_k) \\ &+ \sum_{j'=j+1}^{l} \mathbf{B}_{j'j}(\sum_{k'=1}^{j'-1} \mathbf{B}_{j'k'}^T \boldsymbol{w}_{k'} - \mathbf{B}_{j'} \boldsymbol{w}_{j'}) \end{aligned} \tag{18}$$

In addition to the well-known benefits of skip connections to generalization and trainability, the addition of more feedback connections between layers in our gradients permits information to propagate throughout the network faster, allowing for improved adversarial robustness with fewer iterations.

4. Methods and Results

Our results highlight the improved adversarial robustness of deep pursuit over layered basis pursuit and provide theoretical insights through analysis of several sparse coding metrics, including frame potential, mutual coherence, and reconstruction error.

For the following experiments, we train networks on the CIFAR-10 dataset [13] and use the fast gradient sign method [8] with appropriate values of ϵ to generate our adversarial noise. Adversarial examples are constructed as

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \cdot \operatorname{sign}\Big(\frac{\partial J(\boldsymbol{x}, y)}{\partial \boldsymbol{x}}\Big) \tag{19}$$

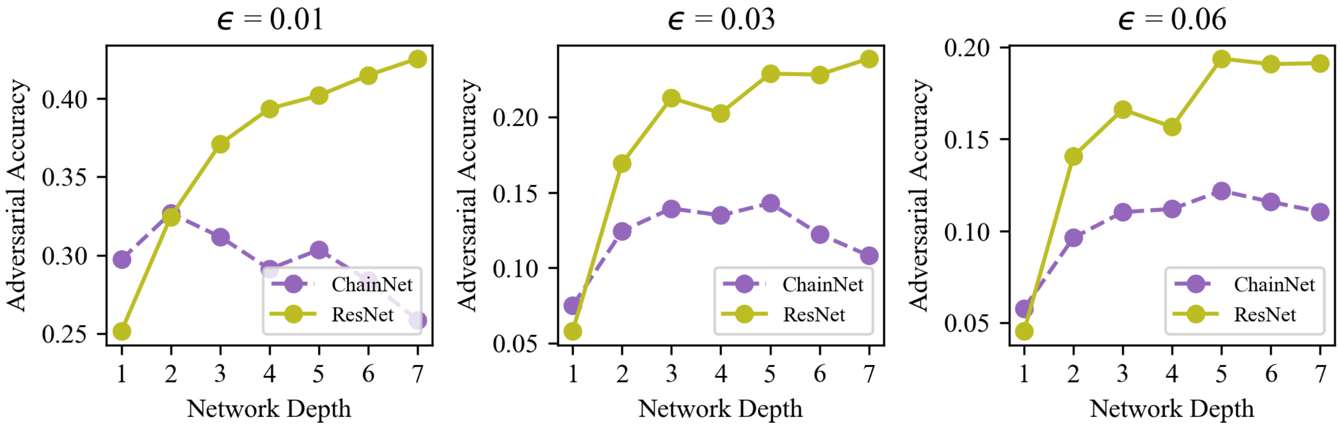

Figure 5. In the feed-forward case, deeper networks with skip connections are, in general, naturally more adversarially robust than shallow networks and those without skip connections. We propose the deep pursuit method to exploit this natural robustness since layered basis pursuit cannot.

by moving each pixel in the direction that would cause the loss to increase the most. We constrain our adversarial perturbations to the ℓ_∞ ball of radius ϵ. On our iterative networks, the adversarial noise is calculated by backpropagating through all the iterations.

We use the modified ResNet architectures from [14] adapted to smaller input resolutions. These are compared against chain networks with residual connections removed, which have the same number of total learned parameters. We also use batch normalization [12] adapted for our recurrent architectures by fixing the scale and offset from the feed-forward initialization for all subsequent iterations. Note that the bias corresponds to the weight of the ℓ_1 penalty λ in Eq. 11, so we constrain it to be non-negative to ensure convexity. All our models in Figure 6 had an accuracy of around 90% on the clean validation set with any variation being negligible.

In our figures, “L-TP” refers to layered thresholding pursuit (feed-forward), “L-BP” refers to layered basis pursuit (local iterations), “DP” refers to deep pursuit (global iterations), and “DP-res” refers to deep pursuit with residual connections. In Figures 6, 9, and 10, each data point represents a unique model trained on T iterations where one iteration indicates a baseline feed-forward network.

4.1. Depth, Skip Connections, and Robustness

From the perspective of sparse approximation, network architectures that can induce global dictionary structures with lower mutual coherence–which is limited by the Welch bound–are less sensitive to input perturbations [14]. Two simple ways of decreasing the Welch bound of an architecture are by making the network deeper or adding skip connections. Figure 5 shows that skip connections improve the adversarial robustness of sufficiently deep feed-forward networks. Furthermore, the residual networks become more robust as we add more layers. These two results motivate developing a new pursuit algorithm that can entertain deeper networks with residual connections.

4.2. Recurrence by Iterative Optimization

Figure 7 shows a sample training curve from a depth-1 residual network with 10 iterations. In all our deep pursuit experiments, we found that most of the improvement in adversarial accuracy came right around when the training accuracy converged. More interestingly, the adversarial accuracy continues to improve well after both training and validation error have converged.

Before moving on to deeper networks with skip connections, we will establish a baseline showing that deep pursuit (global iterations) performs just as well as layered thresholding pursuit (local iterations). Observe in Figure 6 that once we use enough global iterations, we achieve performance comparable to local iterations. We hypothesize that deep pursuit (without skip connections) requires a baseline number of iterations to achieve the full benefit of adversarial robustness because it takes an iteration for information to propagate through a layer before moving to the next.

4.3. Effect of Residual Connections

As neural networks become deeper, skip connections become necessary to achieve state-of-the-art performance [10]. While local layered pursuit could not account for these skip connections, our global deep pursuit incorporates them into the optimization. Figure 6 highlights the benefits of skip connections on adversarial robustness in iterative networks.

In addition to subverting the problem of error accumulation present in layered thresholding pursuit [17], deep pursuit’s compatibility with skip connections are possibly our largest contribution. Not only can we entertain larger, real-

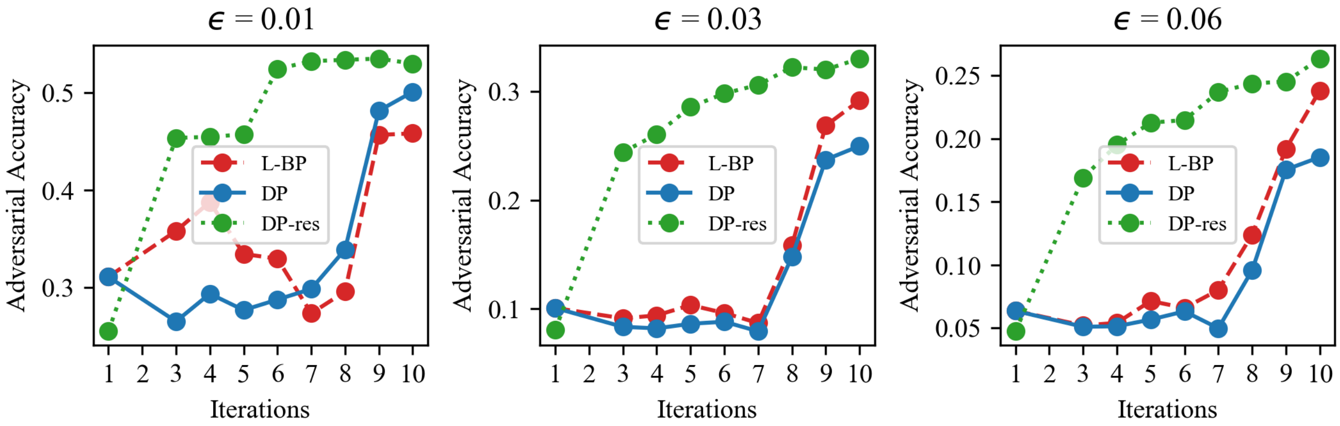

Figure 6. Deep pursuit with residual connections (DP-res) starts to outperform layered basis pursuit (L-BP) almost immediately. Without skip connections, deep pursuit (DP) requires more iterations to propagate information through the layers, so it takes longer to see noticeable improvement. (results from a depth-1 network)

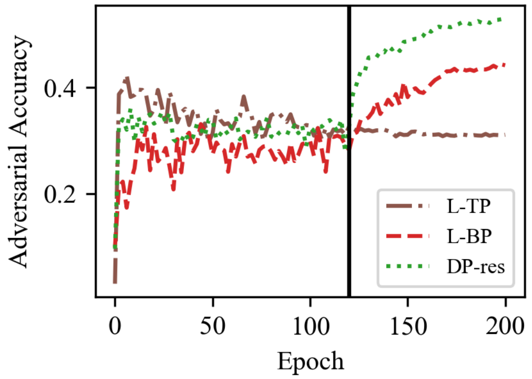

Figure 7. Most of the improvement in adversarial robustness in both layered basis pursuit (L-BP) and deep pursuit (DP) comes after the training error has converged (black vertical line). In the feed-forward case (L-TP), adversarial robustness degrades as training continues.

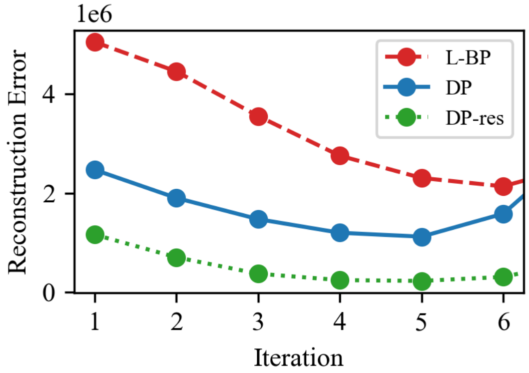

Figure 8. If we analyze the average reconstruction error of each layer per iteration (of a 10 iteration model), we observe that deep pursuit (with and without skip connections) induces more robust embeddings.

world networks, but the addition of skip connections actually makes our algorithm achieve better performance with fewer iterations than on a traditional chain network. We We hypothesize that this is because the skip connections allow information to propagate to more layers per iteration, speeding up the internal optimization.

4.4. Reconstruction Error

As the main objective of sparse approximation algorithms is reconstructing a signal, the reconstruction error of our algorithms should give insight as to why one outperforms another. In our experiments, deep pursuit had, on average, lower reconstruction error per iteration than layered basis pursuit (Figure 8). Adding skip connections further reduced the reconstruction error per iteration.

We also saw that for all three models, the reconstruction error increased over the last few iterations, regardless of the total number. If we only used the conservative Lipschitz constant [4] as the internal step-size for our pursuit algorithms, we would see reconstruction error strictly decrease over the iterations. However, our step size is learned based on the loss of the classification task, explaining the increase in reconstruction error over the last few iterations.

4.5. Deep Welch Bound

As described in [14], mutual coherence (maximum absolute inner product of columns) and frame potential (mean absolute inner product of columns) are measures of the sensitivity of a linear system. Layered basis pursuit and global deep pursuit (without skip connections) have the same global dictionary structure and, therefore, the same

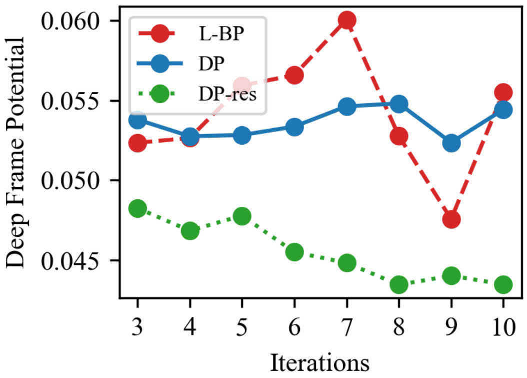

Figure 9. Deep frame potential as described in [14]. As we add more iterations, the deep frame potential of deep pursuit model (with residual connections) decreases, indicating a more robust solution.

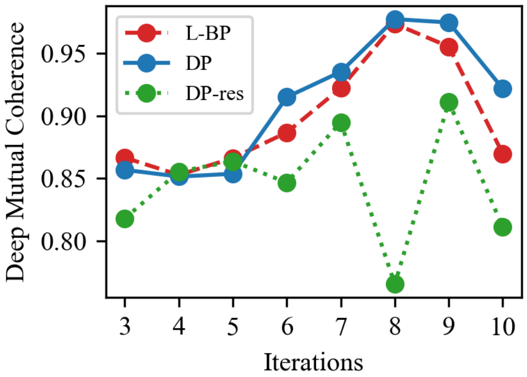

Figure 10. Deep mutual coherence as described in [14]. While a less stable metric than frame potential, we still see that the addition of skip connections (DP-res) induces a lower mutual coherence in our model.

Welch bound. However, since deep pursuit optimizes the whole global dictionary synchronously, the deep frame potential is typically lower than that of layered basis pursuit (Figure 9). Furthermore, we know that skip connections lower the Welch bound, allowing global deep pursuit to have an even smaller deep frame potential and mutual coherence. By comparing Figures 9 and 10 with Figure 6 we see that the deep frame potential and mutual coherence are good predictors of the adversarial robustness of a network.

5. Conclusions

In this work, we extend provably robust sparse approximation techniques to deep neural networks to make them more resistant to adversarial attacks. Previous work reframed each network as a separate sparse coding problem [17], but this technique of layered basis pursuit is prone to error accumulation and cannot entertain modern architectures with skip connections. In contrast, our new method of deep pursuit treats the entire network as a single sparse coding problem to be solved synchronously. As such, our method avoids the issue of error accumulation. Furthermore, by viewing all network parameters as a single structured matrix, our method is easily extended to account for skip connections. Deep pursuit with skip connections consistently out-performs layered basis pursuit and induces a lower deep mutual coherence, which is theoretically tied to adversarial robustness.

Overall, we introduce a strictly architectural method of inducing adversarial robustness in modern neural networks and provide theoretical reasoning for its effectiveness. A theoretical understanding of robust architectures is a significant step towards solving the ever-changing problem of adversarial attacks. As more of modern society continues to rely on computer vision and machine learning, it is critical that we ensure the safety and robustness of these techniques.

6. Acknowledgements

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE1745016 and by the National Science Foundation under Grant No.1925281 along with the CMU Argo AI Center for Autonomous Vehicle Research.

References

[1] Pranjal Awasthi, Himanshu Jain, Ankit Singh Rawat, and Aravindan Vijayaraghavan. Adversarial robustness via robust low rank representations. *Advances in Neural Information Processing Systems*, 33, 2020.

[2] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[3] Jacopo Cavazza, Pietro Morerio, Benjamin Haeffele, Connor Lane, Vittorio Murino, and Rene Vidal. Dropout as a low-rank regularizer for matrix factorization. In *International Conference on Artificial Intelligence and Statistics*, pages 435–444. PMLR, 2018.

[4] Nathaniel Chodosh and Simon Lucey. When to use convolutional neural networks for inverse problems. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8226–8235, 2020.

[5] David L. Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via l1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003.

[6] David L. Donoho, Michael Elad, and Vladimir N. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on Information Theory*, 52(1), 2005.

[7] Zhe Gan, Yen-Chun Chen, Linjie Li, Chen Zhu, Yu Cheng, and Jingjing Liu. Large-scale adversarial training for

vision-and-language representation learning. *arXiv preprint arXiv:2006.06195*, 2020.

[8] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[9] Minghao Guo, Yuzhe Yang, Rui Xu, Ziwei Liu, and Dahua Lin. When nas meets robustness: In search of robust architectures against adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 631–640, 2020.

[10] 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*, pages 770–778, 2016.

[11] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[13] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[14] Calvin Murdock and Simon Lucey. Dataless model selection with the deep frame potential. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11257–11265, 2020.

[15] Vardan Papyan, Yaniv Romano, and Michael Elad. Convolutional neural networks analyzed via convolutional sparse coding. *The Journal of Machine Learning Research*, 18(1):2887–2938, 2017.

[16] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):127–239, 2014.

[17] Yaniv Romano, Aviad Aberdam, Jeremias Sulam, and Michael Elad. Adversarial noise attacks of deep learning architectures: Stability analysis via sparse-modeled signals. *Journal of Mathematical Imaging and Vision*, pages 1–15, 2019.

[18] Kevin Roth, Yannic Kilcher, and Thomas Hofmann. Adversarial training is a form of data-dependent operator norm regularization. *arXiv preprint arXiv:1906.01527*, 2019.

[19] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is robustness the cost of accuracy?–a comprehensive study on the robustness of 18 deep image classification models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 631–648, 2018.

[20] Jeremias Sulam, Ramchandran Muthukumar, and Raman Arora. Adversarial robustness of supervised sparse coding. *Advances in Neural Information Processing Systems*, 33, 2020.

[21] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[22] Yuanhao Xiong and Cho-Jui Hsieh. Improved adversarial training via learned optimizer. *arXiv preprint arXiv:2004.12227*, 2020.

[23] Yangyang Xu and Wotao Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on imaging sciences*, 6(3):1758–1789, 2013.