Advanced ML PA2

Muhammad Huraira Anwar [1] **Adeen Ali Khan** [2]

1. Abstract

Deep neural networks (DNNs) have shown remarkable success across various domains, such as computer vision and natural language processing (NLP), largely due to their universal approximation capabilities. However, the effectiveness of network architectures is closely tied to their structural design, significantly influencing learning efficiency. Traditional network design approaches such as manual design, neural architecture search (NAS), and optimization-based design have each made substantial contributions, exemplified by models like ResNet. Yet, these approaches either demand extensive expertise or function as black-box systems, limiting control over desired properties. In this work, we explore the use of deep unfolding techniques, formulated as a bi-level optimization (BLO), to achieve desirable properties such as sparsity and low-dimensionality in DNNs. We investigate how unfolding the optimization process allows for better control of network properties by leveraging techniques like the Iterative Soft Thresholding Algorithm (ISTA) for l1 norm minimization. Moreover, we extend the framework to incorporate matrix constraints like rank minimization, aiming to capture local and global information in tasks like image processing. We achieve this by implementing learned RPCA (Robust Principal Component Analysis) through a neural network. Through experiments, we evaluate the effectiveness of these optimization-driven designs compared to traditional approaches, providing insights into their ability to handle structured optimization problems and generalize beyond standard neural networks.

2. Introduction

Deep learning models have exhibited remarkable performance across numerous domains, including vision and natural language processing (NLP). However, their generalization capabilities, particularly in terms of robustness and interpretability, remain areas of active research. In many practical applications, neural networks encounter noisy or outlier data, and their ability to handle such scenarios effectively is crucial. This report delves into multiple tasks that explore sparsity, robustness, and optimization techniques, aiming to enhance the understanding and capabilities of deep learning models.

First, we investigate the robustness of different norm-based estimators (L2, L1, and L0) in the presence of noisy observations, evaluating their resilience to outliers in the data. This task sets the foundation for understanding how different loss functions influence the accuracy and robustness of estimators under varying conditions.

Next, we explore the application of sparsity in contrastive models, specifically focusing on CLIP, a state-of-the-art contrastive learning model. We aim to induce sparsity in dense embeddings generated by CLIP for both image and text data to improve interpretability without compromising classification accuracy. Through zero-shot classification experiments, we analyze the trade-off between sparsity and performance, providing insights into how sparse and dense representations compare in terms of memory efficiency and prediction accuracy.

Additionally, we delve into the Iterative Soft Thresholding Algorithm (ISTA), a well-established method for sparse signal recovery. By implementing ISTA and its neural network variant, Learned ISTA (LISTA), we demonstrate the process of recovering sparse signals and compare the performance of traditional and neural network-based approaches. This comparison highlights the effectiveness of LISTA in reducing computational costs while maintaining accuracy in signal recovery tasks.

Lastly, we extend our exploration to matrix-based problems by incorporating rank constraints. Using a low-rank and sparse decomposition approach, we tackle complex data structures, such as images and videos, to achieve robust representations. In doing so, we evaluate the interplay between low-rank assumptions and sparsity, offering a deeper understanding of how these factors influence optimization in high-dimensional data.

This report emphasizes not only the technical implementation of these methods but also the theoretical underpinnings of sparsity, robustness, and optimization in neural networks. By investigating these concepts through the lens of modern machine learning models, we aim to provide insights into how structured optimization approaches can improve model performance, generalization, and interpretability.

3. Methodology

Task 1: Sparsity and Robustness: Mathematical Solution

1. L2 Norm Minimization (Least-Squares)

The **L2 norm minimization** is the least-squares minimization problem. We aim to minimize the following loss function:

$$L_2(z) = \sum_{i=1}^{6} (x_i - z)^2$$

For the given observations $x = [1, 1, 1, 6, 1, 1]$, we can differentiate the loss function with respect to z to find the minimizer:

$$\frac{d}{dz} L_2(z) = \frac{d}{dz} \left(\sum_{i=1}^{6} (x_i - z)^2 \right) = 2 \sum_{i=1}^{6} (z - x_i)$$

Setting the derivative to zero for minimization:

$$2 \sum_{i=1}^{6} (z - x_i) = 0$$

$$\sum_{i=1}^{6} z = \sum_{i=1}^{6} x_i$$

$$6z = 1 + 1 + 1 + 6 + 1 + 1 = 11$$

$$z_{L2} = \frac{11}{6} \approx 1.83$$

To demonstrate that $L_2(z)$ achieves a minimum, we will compute the second derivative and verify that it is greater than zero.

$$\frac{d^2}{dz^2} L_2(z) = \frac{d}{dz} \left(2 \sum_{i=1}^{6} (z - x_i) \right) = 2 \sum_{i=1}^{6} \frac{d}{dz} (z - x_i)$$

Since $\frac{d}{dz}(z - x_i) = 1$, we find:

$$\frac{d^2}{dz^2} L_2(z) = 2 \sum_{i=1}^{6} 1 = 2 \cdot 6 = 12$$

Since $12 > 0$, this indicates that the second derivative is positive, confirming that $L_2(z)$ indeed has a minimum at the point $z_{L2} = \frac{11}{6} \approx 1.83$.

Thus, the **L2 norm estimate** $\hat{z}_{L2}$ is approximately **1.83**.

2. L1 Norm Minimization (Least Absolute Deviations)

The **L1 norm minimization** is the least absolute deviations problem. We aim to minimize the following loss function:

$$L_1(z) = \sum_{i=1}^{6} |x_i - z|$$

The **L1 norm minimizer** in one dimension is the **median** of the observations. The observations $x = [1, 1, 1, 6, 1, 1]$ sorted in increasing order are:

$$x_{\text{sorted}} = [1, 1, 1, 1, 1, 6]$$

The median is the middle value (for an even number of observations, it's the average of the two middle values):

$$z_{L1} = 1$$

Thus, the **L1 norm estimate** $\hat{z}_{L1}$ is **1**.

3. L0 Norm Minimization (Minimizing Non-Zero Differences)

The **L0 norm minimization** counts the number of non-zero differences between the observations x and the estimate z. The goal is to minimize:

$$L_0(z) = \sum_{i=1}^{6} \mathbf{1}(x_i \neq z)$$

Where $\mathbf{1}(x_i \neq z)$ is an indicator function that equals 1 if $x_i \neq z$, and 0 otherwise.

To minimize the number of non-zero differences, we choose the value of z that appears most frequently in the data. In the observations $x = [1, 1, 1, 6, 1, 1]$, the most frequent value is 1 (appearing 5 times out of 6).

Thus, the **L0 norm estimate** $\hat{z}_{L0}$ is **1**.

4. Robustness to Outliers

- **L2 Norm**: The **L2 norm** squares the differences, so large outliers (like 6) have a **disproportionate effect**. As a result, $\hat{z}_{L2} \approx 1.83$, which is biased away from the true value 1 due to the outlier.

- **L1 Norm**: The **L1 norm** only considers absolute differences and does not square them, making it more **robust to outliers**. The L1 minimizer is $\hat{z}_{L1} = 1$, which perfectly captures the majority of the data and ignores the outlier.

- **L0 Norm**: The **L0 norm** completely ignores the magnitude of differences and only counts how many values differ. The

minimizer $\hat{z}_{L0} = 1$ is also robust, as it is unaffected by the outlier.

The L2 norm minimization focused on reducing the sum of squared differences between the estimated value and the data points. This method, however, is particularly sensitive to outliers because squaring the differences gives larger deviations disproportionate weight, causing the estimate to shift toward the outlier. On the other hand, L1 norm minimization reduced the sum of absolute differences, making it less sensitive to large deviations as it does not square the differences. In one-dimensional cases, this method is equivalent to finding the median of the data, which tends to ignore extreme outliers and focus on the central tendency of the majority of values. Finally, the L0 norm minimization simply counted the number of non-zero differences between the estimate and the data points, without considering the magnitude of those differences. This method finds the most frequent value in the dataset, entirely ignoring any outliers by not accounting for the size of deviations, only their presence.

Task 2: Sparsity in Contrastive Models

In this task we explored the CLIP model's dense and sparse embeddings for zero-shot classification, focusing on how sparsity enhances interpretability. Initially, we processed the CIFAR-10 dataset using CLIP's image and text encoders to generate dense embeddings for both images and class labels. These embeddings were normalized to ensure they lay in the same latent space. For classification, we computed cosine similarities between the image and text embeddings, assigning the highest similarity class as the prediction, which served as the baseline for dense embeddings.

To introduce sparsity, we constructed a concept dictionary using CLIP's text encoder to embed both CIFAR-10 classes and additional broader concepts like "machine" and "forest." The concept dictionary was extended beyond CIFAR-10 labels to include verbs (e.g., running, jumping), abstract nouns (e.g., love, fear, happiness), and general nouns (e.g., animal, machine, organism). These additions allowed us to explore how the CLIP model could relate image content to broader, more abstract concepts. For example, verbs like *running* or *jumping* describe dynamic movements often depicted in images, while abstract nouns like *love* or *happiness* let us investigate emotional or psychological associations that images may evoke. General nouns helped capture broader categories overlapping with the CIFAR-10 classes, such as mapping dogs and cats to the concept of "animal" or associating trucks with "machine." This expansion enabled us to observe how well CLIP could associate images not only with specific class labels but also with higher-level or abstract ideas, hypothesizing that images of children or animals might evoke concepts like *playing*, *organism*, or even emotions like *love* or *fear*.

Next, we applied Lasso regression to approximate the dense embeddings with a sparse weight vector, minimizing the difference between the dense image embeddings and the concept dictionary while enforcing sparsity. This method allowed us to generate sparse embeddings, where we could control the level of sparsity by adjusting the regularization parameter. These sparse embeddings were then used for zero-shot classification, and we measured the accuracy for each sparsity level, plotting the relationship between the number of non-zero elements and classification accuracy.

Additionally, we extracted the most important weights from the sparse embeddings for each image, focusing on those with high positive or negative scores, as captured by the absolute value. In this process, the `extract_concepts()` function played a key role, identifying the most important features by considering both strong positive and negative associations. By applying `abs()`, it highlighted the features with the highest magnitude, regardless of sign, as both strong associations and disassociations with a concept were informative. These important weights were then sorted to identify the key concepts or features related to the image, which were mapped back to the corresponding concepts in the dictionary. This approach was crucial for enhancing interpretability, allowing us to understand which visual or conceptual elements the model considered most significant.

Finally, we compared the memory footprints of dense and sparse embeddings. Dense embeddings store full vectors of values, while sparse embeddings store only the non-zero elements and their indices, leading to significant memory savings. This allowed us to evaluate how sparsity affects both performance and efficiency, offering insights into the trade-offs between interpretability and classification accuracy in contrastive models like CLIP.

Task 3: Comparison of ISTA and LISTA

We consider the problem of recovering a sparse vector $z \in \mathbb{R}^{100}$ from noisy measurements. Given a measurement matrix $A \in \mathbb{R}^{100\times100}$ and a noise vector $n \sim \mathcal{N}(0, 0.05^2 I) \in \mathbb{R}^{100}$, the observation equation is defined as:

$$x = Az + n$$

Here, z represents the original sparse vector, and x is the noisy measurement. To control the sparsity of z, we induce it during its generation, and vary the sparsity to observe its effect on the recovery process.

3.1. Implementation of ISTA

An *Iterative Shrinkage-Thresholding Algorithm (ISTA)* was implemented to recover the underlying sparse vector z. The ISTA algorithm was run for a fixed number of iterations,

aiming to minimize the objective function:

$$\min_z \frac{1}{2}\|x - Az\|_2^2 + \lambda\|z\|_1$$

where λ is a regularization parameter that controls the sparsity of the solution. The total loss (comprising the reconstruction loss and sparse loss) was plotted as a function of iterations. The recovered signal $z_{\text{recovered}}$ was compared with the original signal z_{original} using a 2×1 grid visualization.

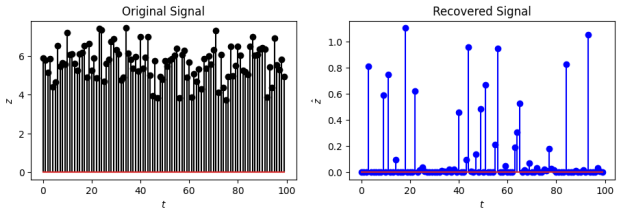


Figure 1. Original Signal vs Noised Signal after 5000 iterations of ISTA at a sparsity level of 0.8 (hence it only recovers 20% of the signal)

3.2. Effect of Sparsity Level on Recovery

To investigate the impact of sparsity on the algorithm's convergence and recovery, the sparsity level of z was varied during its generation. As sparsity increased (i.e. fewer non-zero entries in z), ISTA required fewer iterations to converge. Conversely, as sparsity decreased, the algorithm's recovery accuracy diminished, requiring more iterations.

3.3. Learned ISTA (LISTA)

In a more challenging setup, we assume no knowledge of the measurement matrix A, and the objective is to recover the sparse signal z solely from the noisy observations x. To address this, we unfolded ISTA into a neural network, a method referred to as *Learned ISTA (LISTA)*. LISTA mimics the structure of ISTA but replaces the iterative procedure with learnable blocks of ISTA used as layers, where the depth (number of layers L) was set between 5 and 10.

A dataset of signal pairs was provided, with 9000 training samples and 1000 test samples. The LISTA model was trained on this data to minimize the loss, and the training loss curves were plotted. The test dataset was used to evaluate the model's performance in terms of average loss and average sparsity of the predictions.

The total number of trainable parameters was calculated to compare its complexity with ISTA. Additionally, results on the test dataset were plotted.

3.4. Comparison of ISTA and LISTA

For fair comparison, the same sparsity level in z was provided across both ISTA and LISTA experiments. While ISTA relied on manually tuning parameters like λ, LISTA learned these parameters during training, significantly improving convergence speed and accuracy / final sparsity levels for a fixed number of epochs. The findings indicate that LISTA achieved better reconstruction with fewer iterations due to the optimization of parameters during training.

Task 4: Implementing learned RPCA by unfolding

In this task, we extend our work with vector-based sparsity constraints to address two new challenges: introducing rank constraints into our problem formulation and extending our algorithms to handle matrices. This extension is particularly relevant for processing data types such as images and videos.

We utilize the Moving MNIST dataset for this task. Each sample $X \in \mathbb{R}^{64\times64}$ represents a frame from a video containing two MNIST digits superimposed on a noisy, low-rank background. The provided functions in the accompanying Notebook facilitate the loading and visualization of this dataset.

We design a neural network comprising of total of 8 layers (4 blocks for Sparsity, 4 blocks for Low-Rankness) based on the previously established equations for low-rank and sparse solutions. Each layer of the network is structured to include two branches: one for performing proximal operations for the low-rank component (using Singular Value Decomposition) and the other for the sparse component (using Soft Thresholding). Each block has 3 convolutional layers and a learnable threshold (initialized to 1) used in the proximal operations. This dual-branch architecture enables the model to simultaneously learn both components, adhering to the constraints imposed by the dataset. The proximal operators are as follows:

$$\text{soft}(X, \tau) := \text{sign}(X) \cdot \max(0, X - \tau)$$

$$\text{svt}(X, \kappa) := U \cdot \max(0, \Sigma - \kappa I) \cdot V^T$$

Here are the update rules for the 2 proximal operations:

$$L^{k+1} = \text{svt}\left(C_1^k * L^k + C_2^k * S^k + C_3^k * X, \mu_1\right)$$

$$S^{k+1} = \text{soft}\left(K_1^k * L^k + K_2^k * S^k + K_3^k * X, \mu_2\right)$$

The loss function used for training the network is defined as follows:

$$Loss(L, S) = \text{MSE}(\hat{L}, L) + \text{MSE}(\hat{S}, S)$$

Where:
- $\hat{L}$ is the estimated low-rank component
- $\hat{S}$ is the estimated sparse component

The desired loss function which minimizes the number of singular values of L and minimizes L_0 norm of S, but since both these computations have NP-hard complexity, the loss function we use minimizes the sum mean squared errors between L and $\hat{L}$ and between S and $\hat{S}$, and is an approximation of the desired loss function.

During training, the model is optimized using the defined loss function. We monitor the training process by plotting the total loss as a function of the number of epochs to evaluate convergence and performance.Upon completing the training, we evaluate the model on the test dataset and visualize the results through the following plots:

1. **Low Rank:** A 3D plot showing the layer number and index of singular value on the x and y axes, with the singular values themselves plotted on the z-axis.
2. **Sparse:** A 2D plot illustrating the layer number on the x-axis and the number of sparse values on the y-axis, providing insights into the sparsity characteristics across different layers.

This methodology effectively demonstrates how we can extend the ISTA framework to handle matrix-based data while incorporating rank constraints, facilitating the analysis of complex visual data such as videos from the Moving MNIST dataset, and this whole process entails implementing RPCA (Robust Principal Component Analysis) as a neural network.

Task 5: Implementing learned RPCA with mixed $L_{1,2}$ norm by unfolding

In this task, we extend our work with the same model as previous Task, but optimizing for a different loss function, specifically , instead of minimizing $\|S\|_1$, we minimize a mixed norm i.e. $\|S\|_{1,2}$ using the same dataset as for Task 4, i.e. MovingMNIST.

The proximal operator for soft thresholding in this case changes, so the new proximal operator is:

$$\text{soft}(X, \tau) := \text{X} \cdot \max(0, 1 - \tau/\|X\|_2)$$

The proximal operator for SVD however remains the same:

$$\text{svt}(X, \kappa) := U \cdot \max(0, \Sigma - \kappa I) \cdot V^T$$

The same update rules are applied in this case:

$$L^{k+1} = \text{svt}\left(C_1^k * L^k + C_2^k * S^k + C_3^k * X, \mu_1\right)$$

$$S^{k+1} = \text{soft}\left(K_1^k * L^k + K_2^k * S^k + K_3^k * X, \mu_2\right)$$

The model is trained using the same loss function as before:

$$Loss(L, S) = \text{MSE}(\hat{L}, L) + \text{MSE}(\hat{S}, S)$$

Where:
- $\hat{L}$ is the estimated low-rank component
- $\hat{S}$ is the estimated sparse component

Finally, we compare the obtained results and plots change if you unfolding the 2 networks as well as the difference in proximal operators and their effects.

4. Results

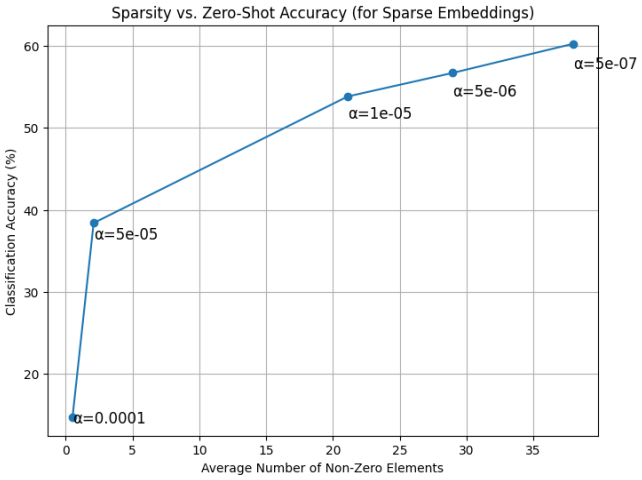
Task2: Sparsity in Contrastive Models

As shown in the plot, the zero shot classification accuracy for sparse embeddings decreases as the average number on non zero element decreases or sparsity level increases. The starting accuracy for sparse embeddings is around **60%**, whereas the accuracy for dense embeddings is around **85%**.

As the average number of non-zero elements in sparse embeddings decreases, the information contained within these embeddings becomes more limited. Sparse embeddings may lose crucial features necessary for accurate classification. Consequently, a lower density of non-zero values can hinder the model's ability to generalize effectively, leading to a decline in zero-shot classification accuracy. In contrast, dense embeddings maintain a richer representation of the data, allowing for better feature extraction and, ultimately, higher accuracy rates.

Moreover, the inherent structure of dense embeddings facilitates the capture of nuanced patterns and relationships within the data. The initial accuracy of approximately **60%** for sparse embeddings, compared to around **85%** for dense embeddings, underscores this point. The dense representations allow the model to leverage a more comprehensive feature set. As sparsity increases, the degradation in accuracy reflects the trade-off between model simplicity and the loss of vital discriminative information.

The memory footprint analysis reveals significant differences between dense and sparse embeddings. The calculation indicates that the dense embeddings require approximately **19.53 MB** of memory, while the latter occupy about **0.04 MB**. This stark contrast is primarily due to the average number of non-zero elements in the sparse embeddings, which is approximately **0.51**. As a result, the sparse representation is highly efficient, consuming minimal memory resources compared to its dense counterpart. The findings demonstrate a remarkable **99.80%** memory savings when utilizing sparse embeddings, highlighting their effectiveness in reducing storage requirements while maintaining essential information.



Task3.3: LISTA The model architecture consisted of **2** LISTA blocks and parameters per block were **10,001** where 10,000 parameters for a fully connected linear layer of input dimension and output dimension of 10,000 without any bias values and a learnable threshold that was used in proximal soft thresholding operator was sampled randomly from [0,1] with uniform distribution. There were a total of **20,002** parameters in the whole network.

We used L1 loss metric to compute loss using training and test time. The training loss converged after **30** epochs to approximately **0.07**. The test loss was around the same value. Average sparsity was **85.25%** which means that only about **15%** elements were non zero.

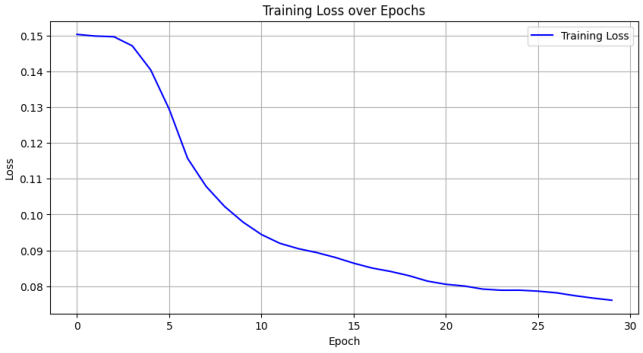


Figure 3. Training loss for LISTA

Task 3.4: Convergence of ISTA vs LISTA on same sparsity level

It is evident from the plots that LISTA converges faster than ISTA on the same signal z. Also, $z_{recovered}$ has sparsity level of **0.65** in LISTA which is way closer to original level of sparsity in z which was **0.66**, whereas $z_{recovered}$ has sparsity level of **0.3** in ISTA.

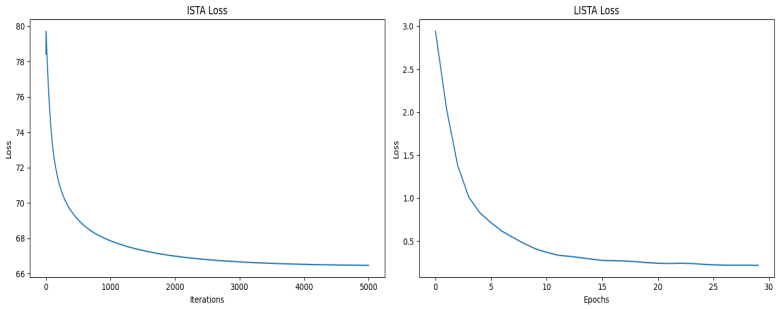


Figure 4. Convergence of ISTA vs LISTA on sparsity level (z) = 0.66

Task 4: Unfolding ISTA with sparse and low rank matrix constraints

As shown in the thresholds graph, the thresholds slowly but surely learn to increase from their initial value, as that helps in minimizing the loss for both tasks, sparsity and low rankness. Because, with a greater threshold, we can push more values to 0, as observable from the proximal operator equations, where τ and κ are the learnable thresholds.

$$\text{soft}(X, \tau) := \text{X} \cdot \max(0, 1 - \tau / \|X\|_2)$$

$$\text{svt}(X, \kappa) := U \cdot \max(0, \Sigma - \kappa I) \cdot V^T$$

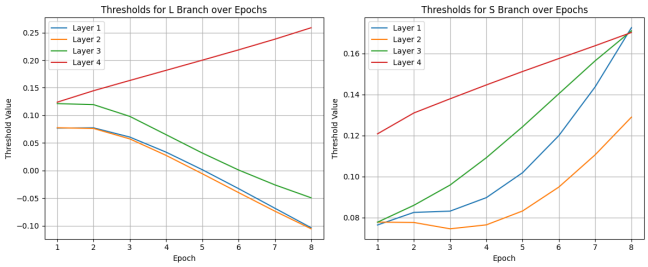


Figure 5. How the thresholds are being learned across layers for learned RPCA with L_1 norm

As seen in the singular values 3D plot, they decrease overtime, but understanding why it maybe increasing in first layer is related to recalling that we are currently minimizing the mean squared error and not singular values themselves directly, Unlike the nuclear norm, which directly penalizes the sum of singular values, MSE does not inherently encourage low-rankness which may result in such a plot, coupled with factors like learnability of thresholds is quite random in earlier layers.

The initial increase in singular values observed in the early layers of the network may be attributed to the network's attempt to capture and represent a wider range of features in the input data. As the optimization process begins, the

network might prioritize expanding its representational capacity by increasing the rank of the low-rank matrix L. This behavior could be seen as the network's strategy to establish a rich feature space before refining and compressing it in later stages of training.

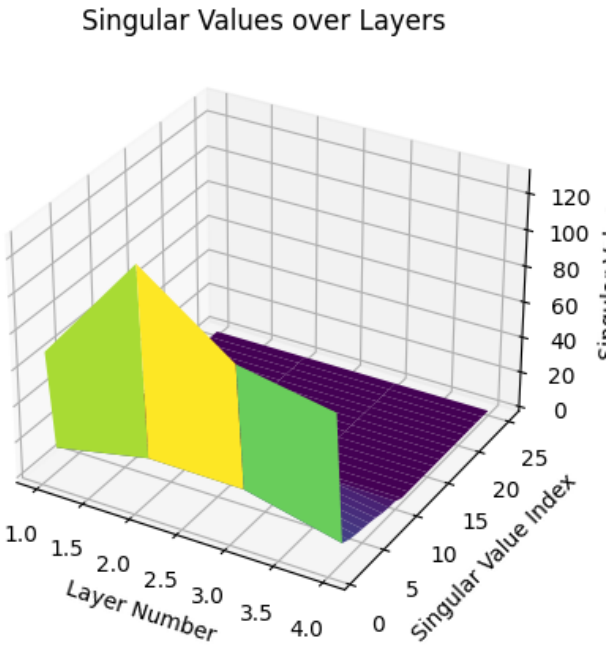


Figure 6. How the singular values are being updated across layers for learned RPCA with L_1 norm

Similarly, for average sparsity plot, the overall trend is of an increase in sparsity, but understanding why it maybe increasing in first layer lies in the loss function where the initial focus on minimizing the reconstruction error rather than enforcing sparsity. When using the L1 norm as a proxy for the L0 norm in the sparse component S, the penalty on non-zero elements is less strict compared to the true L0 norm. This relaxation allows the network to initially distribute small, non-zero values across many elements of S to quickly reduce the MSE between S and $\hat{S}$, resulting in a temporary decrease in sparsity.

Furthermore, this behavior might be exacerbated by the interplay between the low-rank and sparse components. As the low-rank component L initially increases in rank to capture more features, the sparse component S might compensate by becoming less sparse to account for finer details or noise in the data that are not yet efficiently represented by L. This compensation mechanism could lead to a transient phase where sparsity decreases before the optimization process balances the trade-off between reconstruction accuracy and the desired structural properties of both L and S.

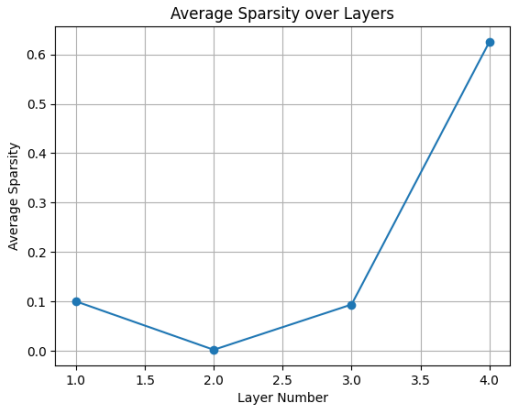


Figure 7. Increase in average sparsity across layers for learned RPCA with L_1 norm

Task 5: Extending unfolded ISTA to learned RPCA with mixed $L_{1,2}$ norm

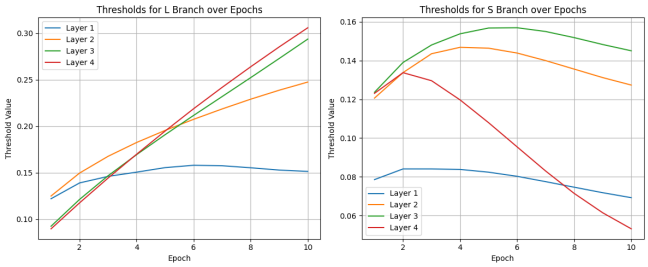


Figure 8. How the thresholds are being learned across layers for learned RPCA with $L_{1,2}$ norm

The linear increase in singular values for the first layer, followed by a linear decrease until the end, can be attributed to the unique properties of the mixed $L_{1,2}$ norm and its corresponding proximal operator. Initially, the network may increase the magnitude of entire columns (or rows) of the matrix to capture important features, as the $L_{1,2}$ norm encourages group sparsity. This behavior leads to an initial increase in singular values as the network attempts to establish a strong basis for representing the data.

The subsequent linear decrease in singular values can be explained by the network's gradual refinement of its representation. As training progresses, the soft thresholding operation in the proximal operator $\text{soft}(X, \tau) := \mathbf{X} \cdot \max(0, 1 - \tau/\|X\|_2)$ begins to have a more pronounced effect. This operation shrinks the magnitude of entire columns (or rows) uniformly, which directly impacts the singular values. The linear nature of this decrease suggests a consistent pruning of less important features across layers, leading to a more compact and efficient representation of the data as the network deepens.

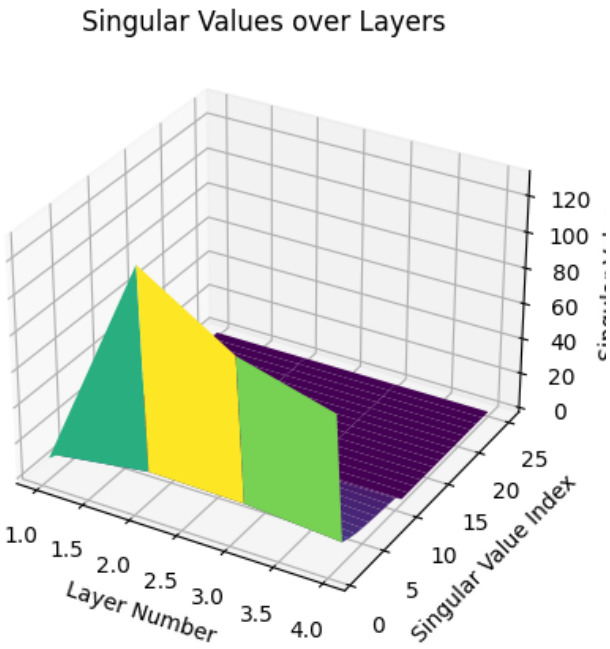


Figure 9. How the singular values are being updated across layers for learned RPCA with L_1 norm

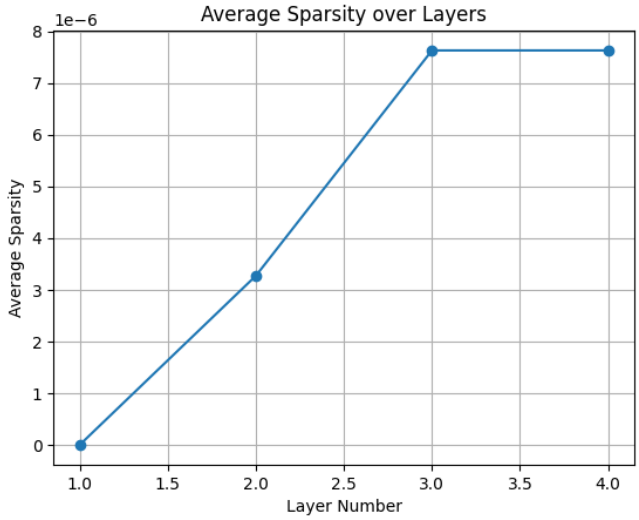


Figure 10. Increase in average sparsity across layers for learned RPCA with $L_{1,2}$ norm

The linear rise in average sparsity until the third layer, followed by a constant sparsity in the last layer, reflects the group sparsity-inducing nature of the mixed $L_{1,2}$ norm. Unlike the L_1 norm, which promotes element-wise sparsity, the $L_{1,2}$ norm encourages entire groups (columns or rows) to become sparse simultaneously. The initial linear increase in sparsity indicates that the network is progressively identifying and emphasizing the most relevant feature groups while suppressing less important ones across the first three layers.

The significantly lower values of average sparsity (ranging from 1e-6 to 8e-6) compared to the RPCA with L_1 norm (0 to 1 scale) highlight the effectiveness of the mixed $L_{1,2}$ norm in achieving group sparsity. This low scale suggests that most groups (columns or rows) are being driven very close to zero, with only a few remaining active. The plateau in sparsity at the last layer may indicate that the network has reached an optimal level of group sparsity, where further sparsification would compromise the model's ability to accurately represent the data. This behavior demonstrates the $L_{1,2}$ norm's capacity to achieve extremely sparse solutions while maintaining the necessary expressiveness in the remaining active groups.

5. Discussion

Task 1: Robustness to Outliers and Sparsity **L1** and **L0** norms are more robust to outliers than the **L2** norm. The **L2 norm** is sensitive to outliers because it squares the differences. The **L1 norm** and **L0 norm** ignore the outlier by focusing on minimizing absolute differences or non-zero differences, respectively.

Task2: Sparsity in Contrastive Models

In Task 2, the analysis of sparse embeddings revealed notable patterns in the association between images and concepts extracted through Lasso regression. Typically, the concept corresponding to the image's class had a high positive weight, such as "airplane" for airplane images, showing strong model confidence. Other relevant concepts, like "sky" and "water," often appeared due to their contextual relationship—airplanes are found in the sky, and the blue color connects sky and water.

Abstract concepts like "love" frequently appeared with high weights, reflecting subjective associations the model learned from diverse data. Both positive and negative weights played a role in interpretability: positive scores indicated strong associations, while negative scores suggested dissociation, giving a richer understanding of how sparse embeddings capture image relationships.

For example, the car image (Fig. 6) is associated with concepts such as 'love,' 'speed,' 'fear,' and 'car,' with both positive and negative weights. This illustrates how the model not only captures the primary object but also links it to abstract or contextual concepts based on visual features and learned associations. The dog image (Fig. 5), on the other

hand, showed high positive weights for 'dog,' 'snow,' 'playing,' and 'jumping,' reflecting the model's ability to connect dynamic activities in the image to relevant concepts.

Comparing CIFAR-10 images to two online images, we found that CIFAR-10 produced more domain-specific concepts like "animal" or "vehicle," whereas the online images showed broader concepts such as "beauty" and "love." This contrast highlighted how sparse embeddings interpret different types of datasets, offering insights into the CLIP model's conceptual representations across varied content.

It relates all kinds of concepts. For example, it relates to the dog *playing* and *jumping* which it is doing, clearly. However, in the image of the car, it even relates to not-so-visible a property it possesses like *speed* or an emotion it may invoke in some due to an accident like *fear*.

Figure 11. Image from outside CIFAR-10 dataset and it's top 4 extracted concepts

Figure 12. Image from CIFAR-10 dataset and it's top 4 extracted concepts

Task3.2: Effect of Sparsity Level on Convergence in ISTA

As sparsity increased (i.e., fewer non-zero entries in z), ISTA required fewer iterations to converge. Conversely, as sparsity decreased, the algorithm's recovery accuracy diminished, requiring more iterations and fine-tuning of lambda. As demonstrated in the figures, the convergence was affected mainly by sparse loss convergence.

The observed phenomenon where increased sparsity in vector z led to faster convergence of ISTA, can be attributed to the inherent characteristics of sparse representations. When the vector contains fewer non-zero entries, the optimization problem becomes simpler, allowing the algorithm to focus on a smaller number of significant components. This results in a more straightforward pathway to convergence, as the algorithm can quickly identify and recover the critical features of the sparse signal. Consequently, the sparse loss converges more rapidly, facilitating a faster overall convergence for the ISTA algorithm.

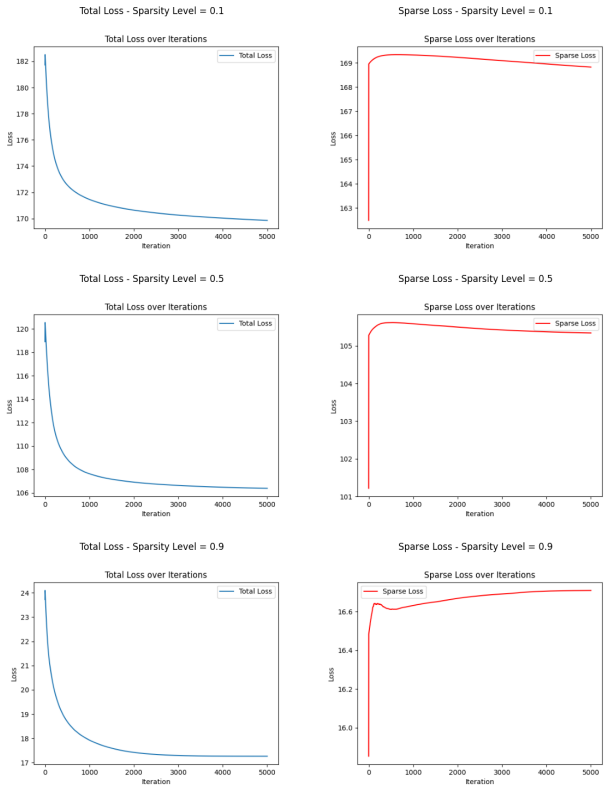


Figure 13. Convergence of Total Loss and Sparse Loss for varying Sparsity Levels of ISTA

Task3.3: LISTA

Using just 2 ISTA blocks in LISTA architecture gave on average **83%** sparsity level. This high sparsity level can be attributed to the inherent structure and functionality of

each block. Each ISTA block is designed to perform a linear transformation followed by a soft-thresholding operation, effectively promoting sparsity in the output signal. The learnable threshold parameter allows the model to adaptively control the level of sparsity based on the input features, enabling it to selectively retain important components while discarding less significant ones.

With two blocks, the architecture strikes a balance between effectively capturing the essential features of the input and enforcing sparsity through the soft-thresholding operation. The linear transformation introduces a level of representation that is sufficiently rich to learn meaningful patterns, while the soft-thresholding actively reduces the number of non-zero entries in the resulting signal. This synergistic effect results in an average sparsity level of **83%**. However, as the number of blocks increases, the compounding effect of the soft-thresholding becomes too restrictive, leading to excessive sparsity and a resultant signal composed entirely of zeros. This illustrates the importance of carefully tuning the number of ISTA blocks to balance sparsity with the retention of meaningful information in the output.

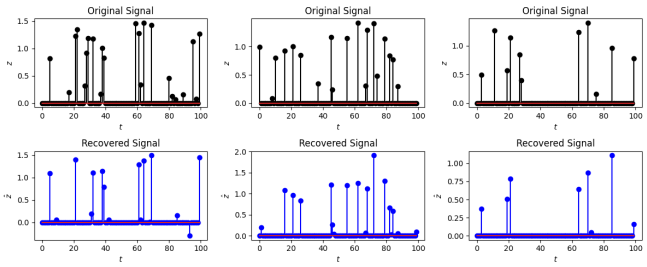


Figure 14. Recovered signals after 85% sparsity for LISTA

Task 4 & 5: Comparison of mixed $L_{1,2}$ norm vs L_1 norm The mixed $L_{1,2}$ norm and the L_1 norm are both commonly used in optimization for promoting sparsity, but they serve different purposes and have distinct advantages. The mixed $L_{1,2}$ norm encourages structured sparsity by promoting the sparsity of entire columns in a matrix, making it particularly useful in applications where features are grouped, such as image processing or multi-task learning. This norm allows for robustness to noise and results in more interpretable models, as entire feature groups can be eliminated. It also effectively reduces dimensionality, enabling joint sparsity in correlated features.

On the other hand, the L_1 norm focuses on individual element sparsity, making it suitable for scenarios where only a few specific features should be non-zero. It is conceptually simpler and widely used in standard regression techniques, such as Lasso. While the L_1 norm effectively prevents overfitting by reducing model complexity, it may lead to less interpretable models in cases where multiple correlated features are relevant, as it does not leverage potential correlations among features.

The choice between these norms should be guided by the specific context of the problem. The mixed $L_{1,2}$ norm is often preferable for data with inherent group structures, while the L_1 norm is advantageous for applications that require individual feature selection. Understanding their respective merits and implications is crucial for effectively utilizing them in optimization tasks related to sparse recovery and feature selection.

In the context of the Moving MNIST dataset, the mixed $L_{1,2}$ norm was expected to outperform the L_1 norm and so it did. The mixed $L_{1,2}$ norm promotes sparsity at the column level, effectively eliminating entire features or digit groups that do not contribute to the reconstruction. This aligns well with the structured nature of the data, allowing for a more interpretable and robust model.

Additionally, the mixed $L_{1,2}$ norm's ability to leverage correlations among grouped features enhances the recovery of meaningful signals from the noisy background. In contrast, the L_1 norm, which focuses on individual feature selection, may struggle to capture the relationships between the digits, potentially leading to suboptimal performance. Thus, for the Moving MNIST dataset, the mixed $L_{1,2}$ norm is more suitable for effectively addressing the task of separating the foreground from the low-rank background.

6. Conclusion

In this report, we explored various optimization-based approaches to enhance the interpretability, robustness, and efficiency of deep learning models, particularly through the lens of sparsity and low-rank assumptions. Our investigation into norm-based estimators revealed that L1 and L0 norms provide stronger resilience to outliers compared to the L2 norm, making them more suitable for noisy data scenarios. We further demonstrated the utility of inducing sparsity in contrastive models, particularly in CLIP, where sparse embeddings offer memory efficiency without significantly compromising classification accuracy. The implementation of ISTA and its neural variant, LISTA, showcased the potential of learned optimization techniques in recovering sparse signals with reduced computational costs. Finally, by extending these optimization principles to matrix-based problems through rank constraints, we successfully achieved robust representations for complex datasets like images and videos. These findings highlight the value of structured optimization techniques in improving model generalization and performance, paving the way for future advancements in both theoretical and practical applications of deep learning.

7. Contributions

Task	Done by
task 1	Adeen
task 2	Adeen
task 3	Huraira
task 4	Huraira
task 5	Huraira

8. References

1. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

2. Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016.

3. Vishal Monga, Yuelong Li, and Yonina C Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38(2):18-44, 2021.

4. Yihua Zhang, Prashant Khanduri, Ioannis Tsaknakis, Yuguang Yao, Mingyi Hong, and Sijia Liu. An introduction to bilevel optimization: Foundations and applications in signal processing and machine learning. *IEEE Signal Processing Magazine*, 41(1):38-59, 2024.

5. Oren Solomon, Regev Cohen, Yi Zhang, Yi Yang, Qiong He, Jianwen Luo, Ruud JG van Sloun, and Yonina C Eldar. Deep unfolded robust PCA with application to clutter suppression in ultrasound. *IEEE transactions on medical imaging*, 39(4):1051-1063, 2019.