CS6302: Advanced Topics In Machine Learning

Deep Unfolding/Unrolling

Ahsan Mir 25100325@lums.edu.pk
Abdulhaseeb Khan 25100077@lums.edu.pk
Aazen Saleem 25100031@lums.edu.pk

Lahore University of Management Sciences

1 Sparsity & Robustness

1.1 Introduction

In this section, we investigate the robustness of various norm minimization techniques when estimating a scalar value from noisy observations, which may include outliers. Our objective is to compare the performance of the ℓ_2, ℓ_1, and ℓ_0 norm minimizers under the influence of such an outlier. This task is particularly relevant in the context of robust machine learning, where sparsity and resistance to outliers play a crucial role in ensuring reliable model performance.

The scalar to be estimated is assumed to have a true value of $z = 1$. Observations are collected through a noisy sensor that, in this case, returns six measurements, $\mathbf{x} = [1, 1, 1, 6, 1, 1]$, where the value 6 represents an outlier. We first estimate z using the least-squares (ℓ_2) norm minimizer, which is highly sensitive to the outlier. We then explore the ℓ_1 norm minimizer, which provides better robustness, and finally, the ℓ_0 norm minimizer, which seeks to capture the sparsest representation.

1.2 Methodology

Here, we detail the step-by-step approach used to estimate the scalar value z using three different norm minimization techniques: the least-squares (L2), the ℓ_1-norm, and the ℓ_0-norm minimizers.

1.2.1 Least-Squares Minimization (ℓ_2 Norm)

We start by estimating z in the least-squares sense, i.e., by minimizing the ℓ_2 norm of the residuals between the observations and the estimate. The corresponding loss function is defined as:

$$L_2(z) = \sum_{i=1}^{6} \|x_i - z\|_2^2$$

Here, x_i represents the individual noisy measurements, where $\mathbf{x} = [1, 1, 1, 6, 1, 1]$. To minimize this loss function, we need to find the value of z that satisfies:

$$\hat{z}_{\ell_2} = \arg\min_z L_2(z)$$

The closed-form solution for the least-squares minimizer is given by the mean of the observed values:

$$\hat{z}_{\ell_2} = \frac{1}{6}\sum_{i=1}^{6} x_i = \frac{1+1+1+6+1+1}{6} = 1.83$$

The solution represents the average of all measurements, including the outlier. This method is highly sensitive to the presence of outliers because the large deviation in the value $x_4 = 6$ pulls the mean upwards, making it less robust.

1.2.2 ℓ_1 Norm Minimization

Next, we estimate z using the ℓ_1-norm minimization technique. The corresponding loss function is formulated as:

$$L_1(z) = \sum_{i=1}^{6} \|x_i - z\|_1$$

Again, x_i represents the measurements. To find the minimizer, we solve:

$$\hat{z}_{\ell_1} = \arg\min_z L_1(z)$$

Unlike the ℓ_2-norm, the ℓ_1-norm minimization is not differentiable at all points. However, it is well-known that the minimizer of the ℓ_1-norm is given by the median of the observations:

$$\hat{z}_{\ell_1} = \text{median}(\mathbf{x}) = \text{median}([1, 1, 1, 6, 1, 1]) = 1$$

The median is much more robust to the presence of outliers, as it does not change significantly when large deviations like $x_4 = 6$ are introduced. Therefore, the ℓ_1-norm minimizer provides a more robust estimate compared to the ℓ_2-norm.

1.2.3 ℓ_0 Norm Minimization

Finally, we estimate z using the ℓ_0-norm minimization technique. The loss function in this case is formulated as:

$$L_0(z) = \sum_{i=1}^{6} \|x_i - z\|_0$$

The ℓ_0-norm counts the number of non-zero residuals, i.e., the number of measurements that do not match the estimate z. To find the minimizer, we solve:

$$\hat{z}_{\ell_0} = \arg\min_z L_0(z)$$

In practice, the ℓ_0-norm minimization seeks the most frequently occurring value, which is the mode of the dataset:

$$\hat{z}_{\ell_0} = \text{mode}(\mathbf{x}) = \text{mode}([1, 1, 1, 6, 1, 1]) = 1$$

Similar to the ℓ_1-norm minimizer, the ℓ_0-norm minimizer is highly robust to outliers. The mode of the data is unaffected by the large deviation introduced by $x_4 = 6$, thus providing a robust estimate.

1.3 Results

We present the results of the three different norm minimization approaches, comparing their respective estimators $\hat{z}$.

1.3.1 Least-Squares Minimization (ℓ_2 Norm)

As derived in the methodology, the least-squares minimizer $\hat{z}_{\ell_2}$ is given by the mean of the dataset:

$$\hat{z}_{\ell_2} = 1.83$$

1.3.2 ℓ_1 Norm Minimization

The ℓ_1-norm minimizer is the median of the dataset:

$$\hat{z}_{\ell_1} = 1$$

1.3.3 ℓ_0 Norm Minimization

The ℓ_0-norm minimizer is the mode of the dataset:

$$\hat{z}_{\ell_0} = 1$$

1.3.4 Visualization

To better understand the impact of the outlier and the different estimators' robustness, we generated a plot showing the data and the estimates. The plot below visually compares the observed values, the outlier at x_4, and the three norm minimizers:

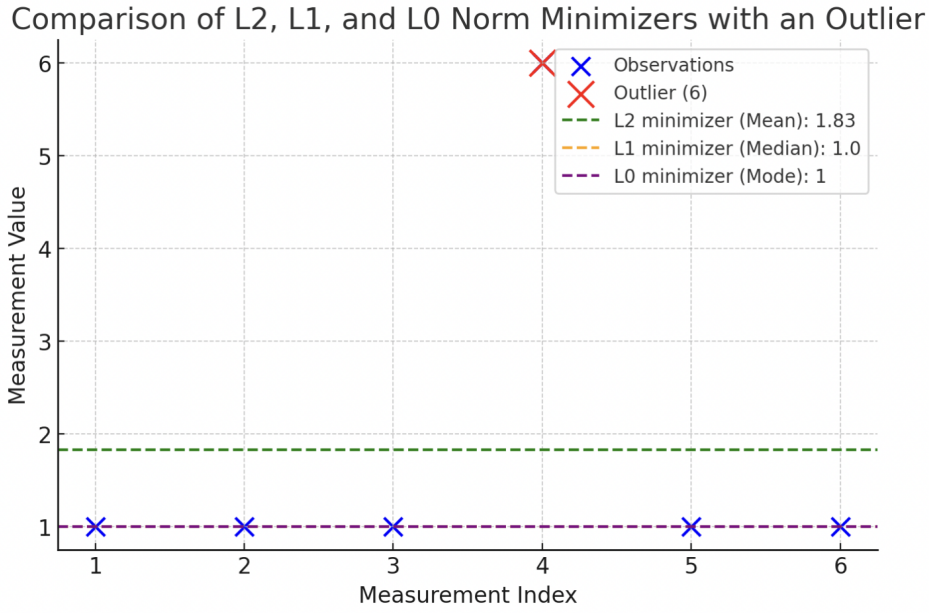

Figure 1: Visualization of Observed Data and Norm Minimizers (ℓ_2, ℓ_1, ℓ_0)

As can be seen in Figure 1, the ℓ_2-norm minimizer (mean) is noticeably influenced by the outlier, while both the ℓ_1-norm (median) and the ℓ_0-norm (mode) remain at the majority value of 1, demonstrating their robustness.

1.4 Discussion

Based on the results, both the ℓ_1-norm and ℓ_0-norm estimators demonstrate robustness in the presence of outliers, while the ℓ_2-norm estimator is sensitive to such deviations. This highlights the importance of choosing the appropriate loss function in practical applications where outliers may occur. In cases where outliers are expected or prevalent, the use of robust estimators like the ℓ_1-norm or ℓ_0-norm minimizers is preferable.

2 Sparsity In Contrastive Models

2.1 Introduction

Despite its impressive performance in zero-shot classification tasks, the interpretability of dense embeddings generated by CLIP remains a challenge, limiting our understanding of the underlying mechanisms that contribute to its classification capabilities. To address this limitation, we investigate the application of sparse signal recovery methods, specifically ℓ_1-relaxation, to derive sparse representations from CLIP's dense embeddings. The objective of this section is to explore how sparsity-inducing techniques can enhance the interpretability of multimodal models without compromising their classification performance. By formulating the problem as an optimization task:

$$\min_{\mathbf{z}} \|\mathbf{x} - \mathbf{A}\mathbf{z}\|_2^2 + \lambda \|\mathbf{z}\|_1$$

where $\mathbf{x} \in \mathbb{R}^d$ represents the dense image embedding, $\mathbf{A} \in \mathbb{R}^{d \times c}$ is the concept dictionary composed of text embeddings, $\mathbf{z} \in \mathbb{R}^c$ is the sparse weight vector, and λ is the regularization parameter controlling sparsity, we aim to recover a sparse vector $\mathbf{z}$ that effectively reconstructs the original embedding $\mathbf{x}$ while promoting sparsity.

To achieve this task, we utilize the CIFAR-10 dataset, a widely recognized benchmark in image classification, to evaluate the efficacy of sparse CLIP representations in a zero-shot classification setting. By comparing the performance of sparse embeddings against their dense counterparts, we seek to understand the trade-offs between sparsity and classification accuracy. Our findings demonstrate that sparse embeddings not only achieve comparable, but in some instances surpass, the classification accuracy of dense embeddings, thereby validating the potential of sparsity-based approaches in enhancing model interpretability and efficiency.

The subsequent subsections of this section delve into the detailed methodology employed for generating and optimizing sparse representations, the experimental results obtained from zero-shot classification tasks, and a comprehensive discussion synthesizing the implications of our findings. Through this exploration, we contribute to the broader discourse on making advanced multimodal models more interpretable and resource-efficient without sacrificing performance.

2.2 Methodology

2.2.1 Zero-Shot Classification with Dense CLIP Embeddings

In this section, we detail the methodology employed to perform zero-shot classification on the CIFAR-10 dataset using dense embeddings generated by a pre-trained CLIP model. Our objective is to leverage the powerful feature representations of CLIP to classify unseen images based solely on the similarity between image and text embeddings.

We utilize the pre-trained CLIP-ViT-Base model which comprises a Vision Transformer (ViT) for image encoding and a Transformer-based text encoder. Both encoders project their respective inputs into a shared latent space of dimensionality $d = 512$, facilitating direct comparison via cosine similarity.

The CIFAR-10 dataset, consisting of 60,000 32×32 color images across 10 distinct classes, serves as our evaluation benchmark. To align with CLIP's input requirements, we apply the following preprocessing steps to the dataset:

1. **Resizing and Cropping:** Images are resized to 224×224 pixels and center-cropped.
2. **Normalization:** Pixel values are normalized using CLIP's predefined mean and standard deviation:

$$\text{Normalized Pixel} = \frac{\text{Pixel} - \mu}{\sigma}$$

where $\mu = (0.48145466, 0.4578275, 0.40821073)$ and $\sigma = (0.26862954, 0.26130258, 0.27577711)$ for the RGB channels.

To generate dense image embeddings, $\mathbf{z}_{\text{img}}$, we pass each preprocessed image through CLIP's vision encoder. For each image $\mathbf{I}$, this embedding is computed as:

$$\mathbf{z}_{\text{img}} = \text{CLIP}_{\text{ViT-B/32}}(\mathbf{I}) \in \mathbb{R}^{512}$$

Each embedding is then normalized to unit length:

$$\mathbf{z}_{\text{img}} \leftarrow \frac{\mathbf{z}_{\text{img}}}{\|\mathbf{z}_{\text{img}}\|_2}$$

To generate text embeddings, $\mathbf{z}_{\text{txt}}$, for each class label, we construct a textual prompt of the form "a photo of a *[classname]*". These prompts are then tokenized and passed through CLIP's text encoder to obtain dense text embeddings:

$$\mathbf{z}_{\text{txt}} = \text{CLIP}_{\text{Text}}(\text{"a photo of a [classname]"}) \in \mathbb{R}^{512}$$

Similarly, each text embedding is then normalized:

$$\mathbf{z}_{\text{txt}} \leftarrow \frac{\mathbf{z}_{\text{txt}}}{\|\mathbf{z}_{\text{txt}}\|_2}$$

To implement the zero-shot classification task, we leverage the cosine similarity between image and text embeddings to assign class labels to unseen images. The process is outlined as follows:

1. **Similarity Computation:** For each test image embedding $\mathbf{z}_{\text{img}}$, we compute the cosine similarity with each class text embedding $\mathbf{z}_{\text{txt}}^{(i)}$:

$$s_i = \mathbf{z}_{\text{img}} \cdot \mathbf{z}_{\text{txt}}^{(i)}, \quad \forall i \in \{1, 2, \ldots, 10\}$$

2. **Prediction Assignment:** Assign the image to the class with the highest similarity score:

$$\hat{y} = \arg\max_i s_i$$

2.2.2 Concept Dictionary Construction and Embedding Alignment

Our goal is to employ sparse signal recovery techniques to enhance the interpretability of dense CLIP image embeddings. The primary objective is to approximate each dense image embedding $\mathbf{z}_{\text{img}} \in \mathbb{R}^d$ using a sparse combination of predefined concept vectors derived from natural language tokens. This process facilitates a more interpretable representation by associating image features with human-understandable concepts.

To construct our concept dictionary $\mathbf{A} \in \mathbb{R}^{d \times c}$, where each column represents a normalized text embedding of a single-word concept, we encompass both the CIFAR-10 class labels and the top 1000 most common concepts extracted from the Flickr30k dataset captions.

To extract concepts from the image captions of FLickr30k, we break down the captions into individual words using tokenization, remove punctuation, numerical values, and stopwords to retain meaningful words, and identify nouns using Part-of-Speech (POS) tagging to ensure that only single-word, descriptive concepts are selected.

Mathematically, let $\mathcal{C}$ denote the set of all extracted nouns from the captions:

$$\mathcal{C} = \{\text{word} \mid \text{word is a noun in captions}\}$$

From the extracted nouns, we select the top 1000 most frequent concepts. Additionally, we incorporate the 10 CIFAR-10 class labels in the dictionary.

$$\mathcal{C}_{\text{top}} = \{\text{Top 1000 nouns from captions}\}$$

$$\mathcal{C}_{\text{final}} = \mathcal{C}_{\text{top}} \cup \{\text{CIFAR-10 classes}\}$$

Finally, we are left with 1003 concepts (after removing duplicates). Next, utilizing CLIP's text encoder, we generate embeddings for each concept word in $\mathcal{C}_{\text{final}}$. For each concept j, we construct a textual prompt of the form "a photo of a *[concept]*," tokenize it, and encode it into the shared latent space:

$$\mathbf{A} = [\mathbf{z}_{\text{txt}_1}, \mathbf{z}_{\text{txt}_2}, \ldots, \mathbf{z}_{\text{txt}_c}] \in \mathbb{R}^{512 \times 1003}$$

where each text embedding $\mathbf{z}_{\text{txt}_j}$ is normalized to unit length:

$$\mathbf{z}_{\text{txt}_j} = \frac{\text{CLIP}_{\text{Text}}(\text{"a photo of a } \textit{[concept]}\text{"})}{\|\text{CLIP}_{\text{Text}}(\text{"a photo of a } \textit{[concept]}\text{"})\|_2}$$

Furthermore, we align the dense image embeddings $\mathbf{z}_{\text{img}} \in \mathbb{R}^{10000 \times 512}$ and the concept dictionary $\mathbf{A} \in \mathbb{R}^{512 \times 1003}$ within the same shared latent space to mitigate modality gaps between image and text embeddings, ensuring that both lie in a comparable semantic space conducive to sparse approximation. The process is outlined below:

1. **Mean Vector Computation:**

We compute the mean vector of the dense image embeddings and the mean vector of the concept dictionary:

$$\boldsymbol{\mu}_{\text{img}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{z}_{\text{img}_i} \in \mathbb{R}^{1 \times 512}$$

$$\boldsymbol{\mu}_c = \frac{1}{c} \sum_{j=1}^{c} \mathbf{A}_{:,j} \in \mathbb{R}^{512 \times 1}$$

where $n = 10,000$ is the number of test images and $c = 1003$ is the number of concepts.

2. **Centering Embeddings:**

Subtract the respective mean vectors to center the embeddings:

$$\mathbf{z}_c = \mathbf{z}_{\text{img}} - \boldsymbol{\mu}_{\text{img}} \in \mathbb{R}^{10000\times 512}$$

$$\mathbf{A}_c = \mathbf{A} - \boldsymbol{\mu}_c \in \mathbb{R}^{512\times 1003}$$

Centering ensures that the embeddings have a mean of zero, facilitating more effective sparse approximation by reducing bias in the data.

3. **Normalization:**

Finally, we normalize both the centered image embeddings and the centered concept dictionary to unit norm along their respective dimensions:

$$\mathbf{z}_{c_{\text{normalized}}} = \frac{\mathbf{z}_c}{\|\mathbf{z}_c\|_2} \in \mathbb{R}^{10000\times 512}$$

$$\mathbf{A}_{c_{\text{normalized}}} = \frac{\mathbf{A}_c}{\|\mathbf{A}_c\|_2} \in \mathbb{R}^{512\times 1003}$$

This normalization places both sets of embeddings on the unit hypersphere, ensuring that cosine similarity metrics are meaningful and that the scales of the embeddings do not disproportionately influence the sparse recovery process.

At this point we have achieved a neatly aligned and normalized framework which can serve as the foundation for inducing sparsity in dense embeddings.

2.2.3 Sparse Representation via Iterative Shrinkage-Thresholding Algorithm (ISTA)

To derive a sparse representation $\mathbf{w} \in \mathbb{R}^c$ for each dense image embedding $\mathbf{z}_c \in \mathbb{R}^d$, we formulate the following optimization problem:

$$\min_{\mathbf{w}} \|\mathbf{z}_c - \mathbf{A}_c\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_1$$

where:

- $\mathbf{z}_c \in \mathbb{R}^{10000\times 512}$ denotes the centered and normalized dense image embeddings for the CIFAR-10 test set.
- $\mathbf{A}_c \in \mathbb{R}^{512\times 1003}$ represents the centered and normalized concept dictionary, with each column corresponding to a concept embedding.
- $\mathbf{w} \in \mathbb{R}^{1003}$ is the sparse weight vector to be determined.
- λ is the regularization parameter that controls the sparsity level of $\mathbf{w}$. We set this to 0.05.

The objective balances the fidelity of the approximation (minimizing the reconstruction error) with the sparsity of the weight vector, promoting interpretability by ensuring that only a few concepts significantly contribute to the representation of each image embedding.

The Iterative Shrinkage-Thresholding Algorithm (ISTA) is an optimization algorithm tailored for solving problems with a smooth differentiable component and a non-smooth regularization term, such as the ℓ_1 norm in our formulation. The algorithm iteratively updates the weight vector $\mathbf{w}$ to converge towards an optimal sparse solution.

1. **Initialization:** Begin with an initial guess for the weight vector, typically $\mathbf{w}^{(0)} = \mathbf{0} \in \mathbb{R}^{1003}$.

2. **Gradient Descent Step:** At each iteration k, compute the gradient of the smooth part of the objective function with respect to $\mathbf{w}$:

$$\nabla f(\mathbf{w}^{(k)}) = 2\mathbf{A}_c^\top(\mathbf{A}_c\mathbf{w}^{(k)} - \mathbf{z}_c)$$

Update the weight vector using a learning rate η (we use 0.01):

$$\mathbf{w}^{(k+1/2)} = \mathbf{w}^{(k)} - \eta\nabla f(\mathbf{w}^{(k)})$$

3. **Soft-Thresholding Step:** Apply the soft-thresholding operator to enforce sparsity:

$$\mathbf{w}^{(k+1)} = \text{sign}(\mathbf{w}^{(k+1/2)}) \odot \max\left(|\mathbf{w}^{(k+1/2)}| - \eta\lambda, 0\right)$$

where $\odot$ denotes element-wise multiplication.

4. **Convergence:** Repeat steps 2 and 3 until convergence criteria are met (e.g., maximum number of iterations or negligible change in $\mathbf{w}$).

For each image embedding $\mathbf{z}_c^{(i)} \in \mathbb{R}^{512}$ in the test set, we solve the optimization problem to obtain the corresponding sparse weight vector $\mathbf{w}^{(i)} \in \mathbb{R}^{1003}$. The sparsity of $\mathbf{w}^{(i)}$ ensures that only a limited number of concepts from the dictionary $\mathbf{A}_c$ are active in reconstructing $\mathbf{z}_c^{(i)}$, thereby enhancing interpretability. Mathematically, the reconstruction of the sparse image embedding is given by:

$$\hat{\mathbf{z}}_{\text{img}}^{(i)} = \mathbf{A}\mathbf{w}^{(i)} + \boldsymbol{\mu}_c$$

where $\boldsymbol{\mu}_c$ is the mean vector of the concept dictionary, reintroduced to maintain the original embedding's mean.

After obtaining the sparse representations $\hat{\mathbf{z}}_{\text{img}}^{(i)}$, we perform zero-shot classification analogous to the dense case:

1. **Cosine Similarity Computation:** Compute the cosine similarity between each reconstructed sparse embedding $\hat{\mathbf{z}}_{\text{img}}^{(i)}$ and the text embeddings of the class labels $\mathbf{z}_{\text{txt}}^{(j)}$:

$$s_{ij} = \frac{\hat{\mathbf{z}}_{\text{img}}^{(i)} \cdot \mathbf{z}_{\text{txt}}^{(j)}}{\|\hat{\mathbf{z}}_{\text{img}}^{(i)}\|_2 \|\mathbf{z}_{\text{txt}}^{(j)}\|_2}, \quad \forall j \in \{1, 2, \ldots, 10\}$$

2. **Class Prediction:** Assign each image to the class with the highest similarity score:

$$\hat{y}^{(i)} = \arg\max_j s_{ij}$$

2.3 Results

2.3.1 Zero-Shot Classification Accuracy

We evaluated the performance of both dense and sparse embeddings in a zero-shot classification task on the CIFAR-10 dataset. The dense embeddings, derived directly from the pre-trained CLIP-ViT-Base model, achieved a classification accuracy of **87.83%**. In contrast, the sparse embeddings, obtained through the Iterative Shrinkage-Thresholding Algorithm (ISTA), achieved a slightly higher accuracy of **88.53%**. This comparative analysis underscores the efficacy of inducing sparsity in embeddings without compromising, and even slightly enhancing, classification performance.

2.3.2 Sparsity Level vs. Classification Accuracy

To understand the relationship between the level of sparsity and classification accuracy, we varied the regularization parameter λ_{reg} in the ISTA optimization process. The hyperparameters were defined as follows:

- $\lambda_{\text{reg}} \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 5\}$
- Number of Iterations: 200
- Learning Rate: 0.01

The results are summarized in Table 1.

Table 1: Sparsity Level and Corresponding Zero-Shot Classification Accuracy for Various λ_{reg} Values

λ_{reg}	**Average Non-Zero Weights per Image**	**Classification Accuracy (%)**
0.01	696.90	88.65
0.02	476.53	88.60
0.05	215.27	88.53
0.10	84.69	87.80
0.20	18.54	85.23
0.50	1.04	48.88
1.00	0.00	10.00
5.00	0.00	10.00

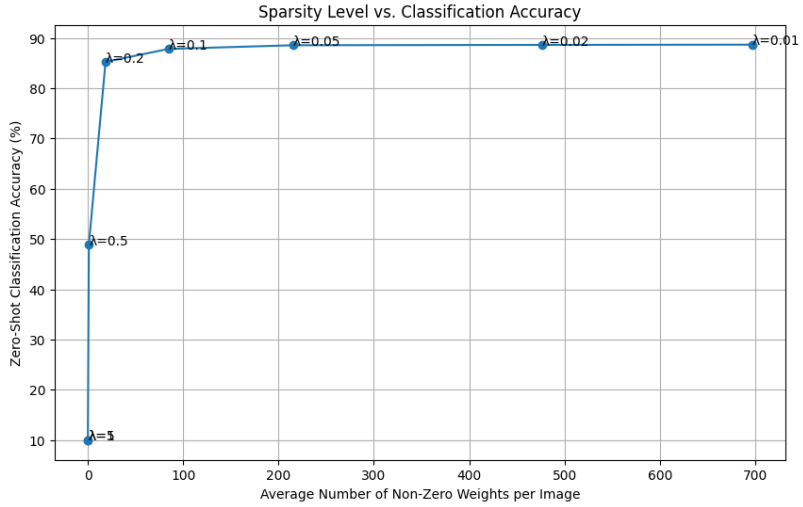

Figure 2: Relationship Between Sparsity Level and Classification Accuracy

The plot in Figure 2 illustrates the inverse relationship between the regularization parameter λ_{reg} and the average number of non-zero weights per image. As λ_{reg} increases, sparsity is induced more aggressively, leading to a reduction in the number of active weights. Correspondingly, classification accuracy tends to decrease, especially at higher λ_{reg} values where sparsity becomes excessive, resulting in significant performance degradation until the model eventually makes predictions based purely on chance (10%).

2.3.3 Memory Footprint Comparison

An essential advantage of sparse embeddings lies in their reduced memory footprint compared to dense embeddings. Table 2 presents a comparison of memory usage between dense and sparse embeddings:

Table 2: Memory Footprint of Dense vs. Sparse Embeddings

Embedding Type	**Shape**	**Memory Footprint (MB)**
Dense	10000×512	19.53
Sparse	10000×1003	8.21

The dense embeddings occupy approximately **19.53 MB**, while the sparse embeddings require significantly less memory, approximately **8.21 MB**. This reduction is attributable to the sparsity of the weight vectors, where only a fraction of the weights are non-zero and thus need to be stored. The decreased memory footprint not only facilitates storage efficiency but also enhances computational speed during inference by minimizing the number of active computations.

2.3.4 Active Concepts Analysis

Understanding which concepts are most active in the sparse embeddings provides valuable insights into the interpretability of the model. Table 3 lists the top 20 most active concepts along with their non-zero counts across the dataset.

Table 3: Top 20 Most Active Concepts in Sparse Embeddings

Rank	Concept (Non-Zero Count)
1	cat (6182)
2	animals (5802)
3	deer (5767)
4	horses (5409)
5	horse (5135)
6	ship (5081)
7	frog (4891)
8	dog (4883)
9	vehicles (4811)
10	vehicle (4606)
11	works (4530)
12	farm (4448)
13	driver (4381)
14	video (4371)
15	pigeons (4332)
16	bat (4331)
17	crew (4302)
18	puppies (4290)
19	jet (4289)
20	job (4282)

The predominance of concepts such as **cat**, **animals**, **deer**, and **dog** aligns well with the CIFAR-10 classes, indicating that the sparse representations effectively capture and emphasize relevant semantic information. This selective activation of concepts enhances the interpretability of the embeddings by linking image features to meaningful, human-understandable concepts.

To further elucidate the relationship between active concepts and image content, we present qualitative examples showcasing images alongside their top contributing concepts.

Figure 3: Sample Image with Active Concept: *cat*

Figure 4: Sample Image with Active Concept: *deer*

2.4 Discussion

2.4.1 Performance Comparison

The sparse embeddings, achieved through the ISTA optimization process, not only matched but slightly surpassed the performance of dense embeddings, achieving an accuracy of **88.53%** compared to the dense embeddings' **87.83%**. This marginal improvement suggests that inducing sparsity can enhance the model's ability to focus on the most salient and relevant concepts, thereby potentially reducing noise and overfitting associated with dense representations.

2.4.2 Sparsity-Accuracy Trade-Off

Our exploration of various λ_{reg} values illuminated a clear trade-off between sparsity and classification accuracy:

- **Low λ_{reg} (e.g., 0.01, 0.02):** These values result in high sparsity levels, with an average of approximately 696.90 and 476.53 non-zero weights per image, respectively. Interestingly, even with significant sparsity, the classification accuracy remains high (**88.65%** and **88.60%**).
- **Moderate λ_{reg} (e.g., 0.05, 0.1):** At $\lambda_{\text{reg}} = 0.05$, the average number of non-zero weights drops to 215.27, with the classification accuracy maintaining at **88.53%**. Further increasing λ_{reg} to 0.1 reduces non-zero weights to 84.69, with a slight decrease in accuracy to **87.80%**.
- **High λ_{reg} (e.g., 0.2, 0.5, 1, 5):** Elevated values of λ_{reg} induce extreme sparsity, reducing the average number of non-zero weights to as low as 18.54 and even zero. This precipitous sparsity leads to significant declines in classification accuracy, plummeting to **85.23%** at $\lambda_{\text{reg}} = 0.2$ and further to **10.00%** at $\lambda_{\text{reg}} = 1$ and 5.

These observations highlight that moderate levels of sparsity can be achieved without detrimental effects on classification performance. However, excessive sparsity severely compromises the model's ability to accurately classify images, underscoring the importance of carefully tuning the regularization parameter.

2.4.3 Memory and Computational Efficiency

The transition from dense to sparse embeddings offers substantial benefits in terms of memory usage and computational efficiency. In terms of memory footprint, dense embeddings occupy approximately **19.53 MB**, whereas sparse embeddings require only **8.21 MB**. This reduction is significant, especially when scaling to larger datasets or deploying models on resource-constrained devices. Furthermore, computationally speaking, sparse representations inherently involve fewer active computations due to the reduced number of non-zero weights. This leads to faster inference times and lower energy consumption.

2.4.4 Interpretability Through Active Concepts

The analysis of active concepts in sparse embeddings reveals that the model predominantly leverages semantically meaningful and relevant concepts corresponding to the CIFAR-10 classes. The top active concepts, such as *animals* and *vehicles*, align closely with the categories present in the CIFAR-10 dataset. This alignment indicates that the sparse recovery process effectively identifies and emphasizes the most pertinent concepts for image classification.

2.5 Conclusion

In this section, we have effectively demonstrated that inducing sparsity in dense CLIP embeddings through ISTA not only maintains but slightly enhances zero-shot classification performance on the CIFAR-10 dataset. The sparse embeddings achieve comparable accuracy with a substantially reduced memory footprint, highlighting their potential for efficient and interpretable multimodal representations. Furthermore, the active concepts analysis affirms that sparse representations effectively capture and emphasize relevant semantic information, thereby enhancing model transparency. Future work should delve deeper into optimizing sparsity parameters, expanding concept dictionaries, and exploring advanced sparsity-inducing techniques to further bolster the efficacy and applicability of sparse contrastive models.