A sparse null code emerges in deep neural networks

Brian S. Robinson[1], **Nathan Drenkow**[1], **Colin Conwell**[2], **and Michael F. Bonner**[2]

[1] Research and Exploratory Development Department, Johns Hopkins University Applied Physics Laboratory, Laurel, MD
[2] Department of Cognitive Science , Johns Hopkins University, Baltimore, MD
[1]{ brian.robinson, nathan.drenkow } @jhuapl.edu
[2]{ colinconwell, mfbonner } @jhu.edu

Editors: Marco Fumero, Emanuele Rodolà, Clementine Domine, Francesco Locatello, Gintare Karolina Dziugaite, Mathilde Caron

Abstract

The internal representations of deep vision models are often assumed to encode specific image features, such as contours, textures, and object parts. However, it is possible for deep networks to learn highly abstract representations that may not be linked to any specific image feature. Here we present evidence for one such abstract representation in transformers and modern convolutional architectures that appears to serve as a null code, indicating image regions that are non-diagnostic for the object class. These null codes are both statistically and qualitatively distinct from the more commonly reported feature-related codes of vision models. Specifically, these null codes have several distinct characteristics: they are highly sparse, they have a single unique activation pattern for each network, they emerge abruptly at intermediate network depths, and they are activated in a feature-independent manner by weakly informative image regions, such as backgrounds. Together, these findings reveal a new class of highly abstract representations in deep vision models: sparse null codes that seem to indicate the absence of features rather than serving as feature detectors.

1 Introduction

When seeking to interpret the internal representations of deep vision models, it is often implicitly assumed that the representations encode image features, such as edges, textures, shapes, and objects (34; 26; 4; 25; 3). This assumption underlies many common approaches for neural network interpretation, such as the selection of strongly activating natural images, 2D tSNE embeddings of image representations, activation maximization through image synthesis, and saliency maps.

However, an alternative possibility is that some aspects of neural network representation encode highly abstract information that is not linked to any specific image feature and would, thus, appear to have incoherent selectivity when examined with standard visualization methods. Here we present evidence for one such abstract representation in deep vision models. Specifically, by examining vision transformers and a modern convolutional architecture, we reveal a class of representations that respond in a feature-independent manner to image regions that are weakly informative for identifying the central object category, such as backgrounds and large homogeneous patches of texture. We refer to this class of representations as "null codes" because they appear to indicate non-diagnostic parts of the image. We also demonstrate these null codes are statistically distinct from other network representations due to their characteristic high activation norms and sparsity.

These sparse null codes constitute a qualitatively different phenomenon than the commonly observed image-feature representations reported in previous work, and they motivate a new perspective on the internal representations of vision models and methods for network interpretation.

2 Approach

In the analysis of this work, we inspect the embeddings of a range of neural networks, including three different transformer architectures and a recent performant convolutional neural network. The three different models with transformer architectures utilized for analysis span differences in training and scale: 1) an original ViT model (Base/16) with patch-wise image tokenization trained on ImageNet (9), 2) the same ViT model trained with Contrastive Language-Image Pretraining (CLIP) (29), and 3) Compact Convolutional Transformer, a more parameter-efficient model with a convolutional tokenizer before the transformer layers and sequence pooling as a final operation before classification (CCT-14/7$\times$2) (13). The convolutional neural network examined here is ConvNext, which modernizes a standard ResNet (14) toward the design of a transformer and achieves competitive accuracy with modern vision transformers networks (22). For analyzing the intermediate embedding of these networks, we utilize image probes from the Microsoft COCO dataset (21).

In transformer networks, we examine intermediate embeddings, $x \in \mathbb{R}^{D,N}$, with N tokens and D dimensions. Our analysis characterizes each feature vector with dimension D, across each image probe, token, and layer. Note that for the ViT and ViT-CLIP networks that have a class input token, this class token was not used in our analyses. Summary statistics for the ViT-CLIP characterization were computed on 500 different images. However, the observed trends were identical to those for the first 24 images (which already contain 56,448 individual features vectors across images, layers, and tokens). For expediency, subsequent analyses on different networks are performed with the first 24 images which entail an analysis of over 50,000 feature vectors for each network model.

We measure the sparsity of activation vectors using a standard measure known as kurtosis, k (27; 15), which characterizes how heavy-tailed a distribution is by measuring the fourth moment relative to the variance squared, with $k > 0$ for a distribution that is sparser or has a heavier tailed distribution than a Gaussian distribution. Note that kurtosis is a relative measure of sparsity as the feature vectors do not have dimensions with values exactly equal to zero.

As opposed to the transformer network, which has the same number of tokens and dimensionality throughout all layers, the ConvNext architecture has four stages with varying feature map size and number of channels. However, the third stage is a natural choice to analyze in detail for comparison because 1) the majority of the network's blocks are at this stage (27 as opposed to 3 blocks in the other stages) and 2) the 14$\times$14 size of the feature maps corresponds to the same size and image locations as the 196 tokens analyzed in the transformer networks. The feature maps have shape $D \times 14 \times 14$ and feature vectors can be comparably used for analysis with all of the channels at a specific location (which contain D elements). The dimensionality of the ConvNext feature map embeddings in each block can thus be similarly reshaped to $D \times N$, where D is the number of channels, N is the number of feature map locations.

3 Observation of a sparse null code

3.1 Main Observations

In our analysis of feature vectors across a range of network variants, we denote five repeated and previously unreported observations in relation to a sparse null code:

1. Sparse patterns emerge abruptly at an intermediate layer depth.
2. There is a bimodal distribution of sparsity across feature vectors.
3. Sparse patterns are maintained at specific locations across layers and correspond to backgrounds or weakly informative portions of images.
4. The sparsest feature vectors for each image presentation have approximately the same activation pattern or "sparse code".
5. *All* of the analyzed highly sparse feature vectors have high similarity with the sparse code.

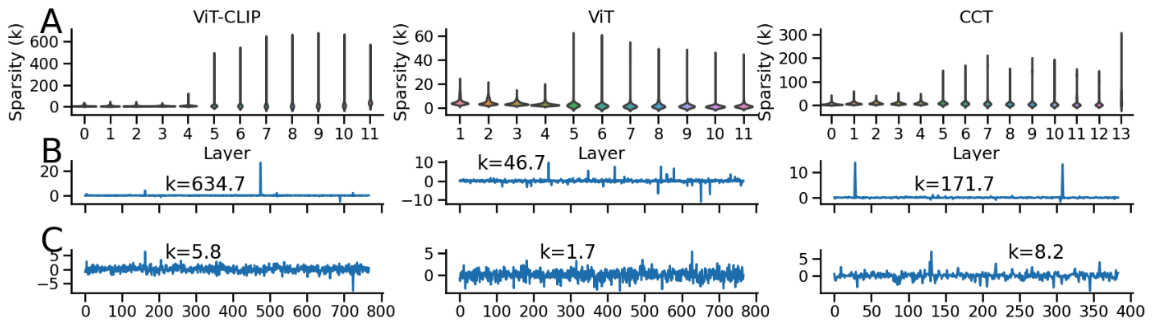

Figure 1: A subset of highly sparse feature vectors emerge in transformer networks. A. The distribution of kurtosis sparsity measurements for each token across 24 presentations for the three transformer model architectures evaluated. B-C. Activation patterns (Z-score) from the sparsest and median sparsity feature vectors for each model. Sparsity measured with kurtosis (k) is displayed for each pattern to visualize how kurtosis relates to sparsest and median feature vector distributions of each model.

3.2 Sparse codes in network variants

Across all three networks with a transformer architecture, a stark increase in measured sparsity is observed for a subset of feature vectors (Fig 1). By visual inspection, many of the feature vectors have sparse-like activation patterns (Fig 1 B-C), with most dimensions close to zero and a subset of dimensions with much larger magnitudes.

Our most in depth characterization is with the ViT-CLIP model, where we characterize these sparse feature vectors for individual images (Fig 2) as well as in aggregate across 500 different images (Fig 4). By visual inspection, we can see that these sparse patterns are maintained at specific tokens across layers (Fig 2B) and that these tokens correspond to background or weakly-informative portions of the input images (Fig 2C). Perhaps most surprisingly, however, is that the sparsest feature vectors for each image presentation appear to have the same activation pattern or "code" (Fig 2D).

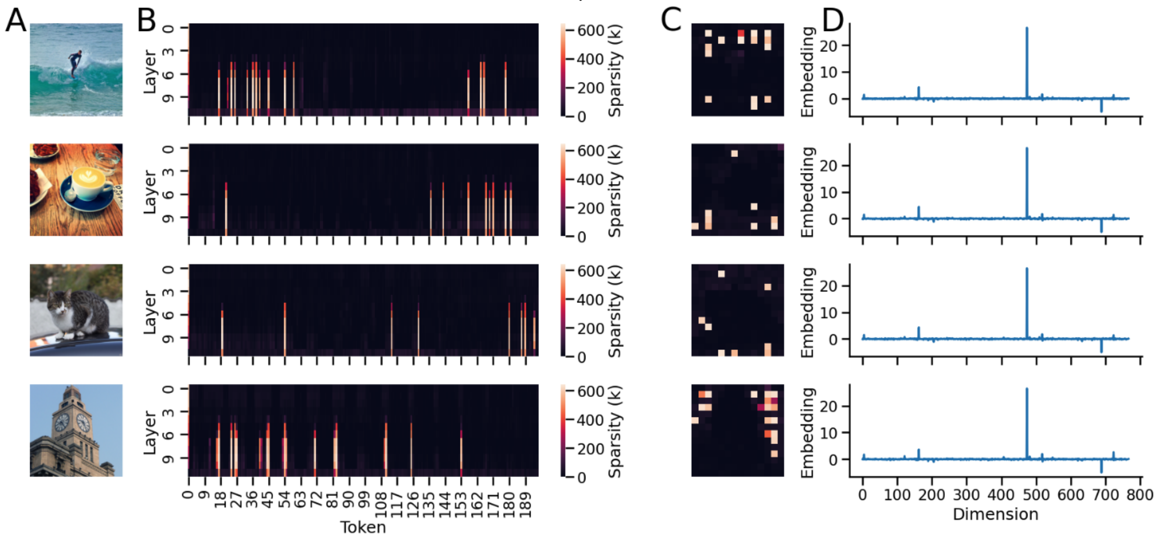

Figure 2: In ViT-CLIP, a repeated sparse code is observed in non-diagnostic token locations. A. Example presentation images from center-cropped MSCOCO dataset used as network probes for each row. B. Sparse patterns are maintained at specific tokens across layers. C. At an intermediate depth of layer 9, the 196 tokens which correspond to image patch inputs are reshaped to a 14×14 matrix. D. The Z-scored feature vector corresponding to the sparsest activation pattern is almost identical by visual inspection between images.

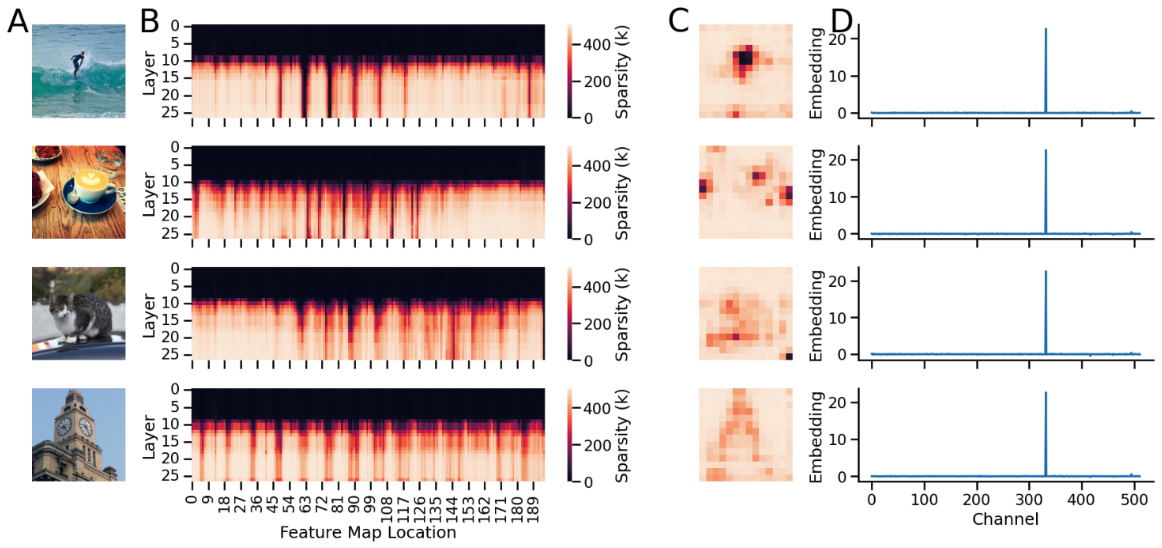

Figure 3: Feature vectors in ConvNext share similar overall observed sparsity properties as seen in the Transformer networks where 1) sparse patterns emerge abruptly at an intermediate layer depth, 2) these sparse patterns are maintained at specific locations across layers, 3) sparse feature vectors correspond to background or non-salient portions of the input images, and 4) the sparsest feature vectors for each image presentation appears to have the same activation pattern or "code". A-D. As in Fig. 2

The feature vectors in ConvNext share similar overall observed sparsity properties (Fig 3) as seen in the transformer networks. Notable differences include the increased prevalence of these sparse feature vectors and how they seem to emerge at all background locations. The repeated sparse code is also mostly confined to a single channel (Fig 5).

It is not only the sparsest feature vectors though, for each image that have the same pattern. When analyzed over 500 image presentations for ViT-CLIP, all highly sparse activation patterns appear to resemble this code (Fig 4). An additional characteristic of these feature vectors is that their sparsity has a distinctly bimodal distribution (Fig 4A). Furthermore, because all of the sparsest feature vectors for each image are almost identical (average pair-wise correlation coefficient of 0.996), we can use the average of these sparsest feature vectors as the sparse code for downstream analysis.

In ConvNext (Fig 5) as well as the ViT and CCT transformer model architectures (Fig A.1), similar trends are observed as with ViT-CLIP, with a bimodal distribution of sparsity in feature vectors, high similarity between the sparsest codes, and all of the highly sparse feature vectors having high similarity with the sparse code. For ViT and CCT, there is slightly more variability in the range of sparsity values and the degree to which the sparsest feature vectors are all approximately the same (0.927 and 0.979 average pair-wise correlation coefficient for ViT and CCT respectively as compared to 0.996 for ViT-CLIP). Note that for ViT and CCT, the first and last layers are excluded from this characterization. Overall, between analyzed transformer and convolutional models, we find similar trends in the emergence of a repeated sparse code as outlined in Section 3.1.

3.3 Characterization of the sparse code

To investigate how these sparse codes emerge and are utilized we characterize how model weights relate to these sparse codes across network depth. Our approach focuses on the measuring the similarity of the input and output weights of each component in each layer to the sparse code. The detailed approach and results for how these sparse codes relate to input and output network weights for each component in Transformer and ConvNext models is outlined in A.2, but there are a few high-level properties that describe how these sparse codes emerge and are utilized. First and foremost, we find evidence that these sparse codes are unlikely to emerge via an isolated anomaly in network weights, as there are consistent properties across multiple layers for how network weights in each component are tuned to input and output these sparse codes. Additionally, for the input weights to the multi-headed self-attention (MHSA) in transformers, the weights which are utilized in the

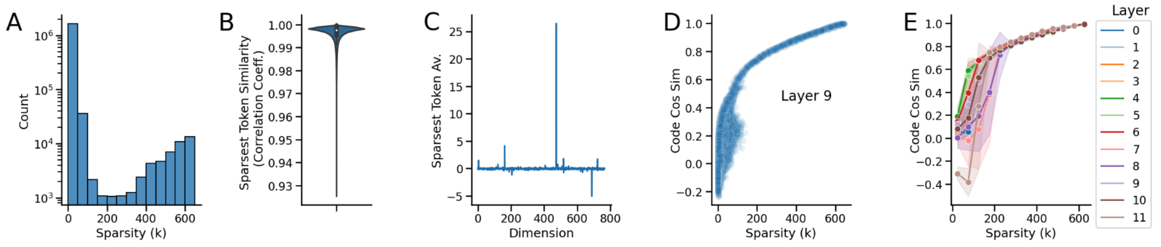

Figure 4: All of the highly sparse feature vectors correspond to the same sparse code in ViT-CLIP across 500 input images. A. The measured sparsity for each feature vector across all layers has a bimodal distribution. B. For every pair-wise combination of the sparsest feature vector in each image presentation, there is a high Pearson's correlation coefficient value, with an average of 0.996. C. The average of all of the highly correlated sparsest feature vectors (Z-scored) is used as the "sparse code" for subsequent analysis. D. In a representative layer (layer 9), cosine similarity is shown between every feature vector across all probe images and the sparse code shown in C. E. As in D, but across all layers, where the average and standard deviation is shown for each of the binned sparsity values from A

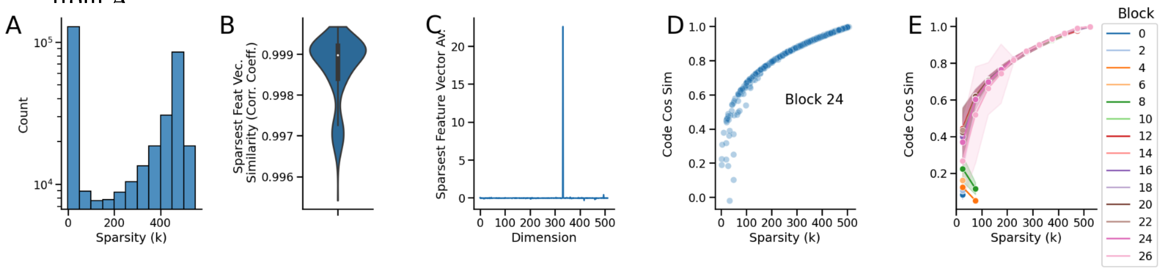

Figure 5: In the ConvNext model, similar trends are observed as with Transformer models with a bimodal distribution of sparsity in feature vectors, high similarity between the sparsest codes, and all of the highly sparse feature vectors having high similarity with the sparse code. A-E. As in Fig 4 with 24 input images.

nonlinear calculation of attention (*query* and *key* weights) are more tuned to the sparse code than the weights that are only involved in a linear transformation (*value* weights). In both transformer and convolutional networks, input weights of each component become more tuned to the sparse null code in final layers or blocks. We additionally investigate whether sparse codes interact with isolated dimensions in network layer components. For the majority of components, we find that this is not the case, as most network weights have vectors which contain a mix of positive and negative tuning in relation to these sparse null codes, suggesting a distributed interaction of these sparse codes on network activity. An exception to this is observed in the output weights of the ViT-CLIP MLP, where a small subset of weight dimensions are highly correlated with the sparse code, suggesting an isolated subset of dimensions in the MLP can play an out-sized role in generating these sparse codes. Overall, these properties describe systematic trends in how the sparse codes relate to input and output weights, yielding insight into how these sparse codes emerge and are utilized across network components.

We observe additional properties of the sparse code that are unique to specific investigated models. While both ConvNext and ViT-CLIP both have an overall bimodal distribution of sparsity in feature vectors across all layers and blocks, ConvNext has a unimodal distribution in each block and ViT-CLiP maintains a bimodal distribution within each individual layer (Fig 6A-B). An additional direct relationship is observed between the magnitude of each weight vector and the level of sparsity for ViT-CLIP, which is maintained between layers 6C-D). While an overall linear relationship between magnitude and sparsity is also observed for ConvNext, the relation is more apparent in ViT-CLIP. These observed differences as well as the variation in levels of sparsity between models motivates future investigation.

As outlined above, sparse codes are observed at specific locations, but it is unclear to what extent these codes relate to the remaining feature vectors. To investigate this, the sparse code calculated for each model is compared to the principal components of all of the feature vectors for each layer across 500 probe images. As shown in Fig 7, for all models, the sparse code has a high cosine similarity to the first principal component of the feature vectors in most layers after the sparse code emerges.

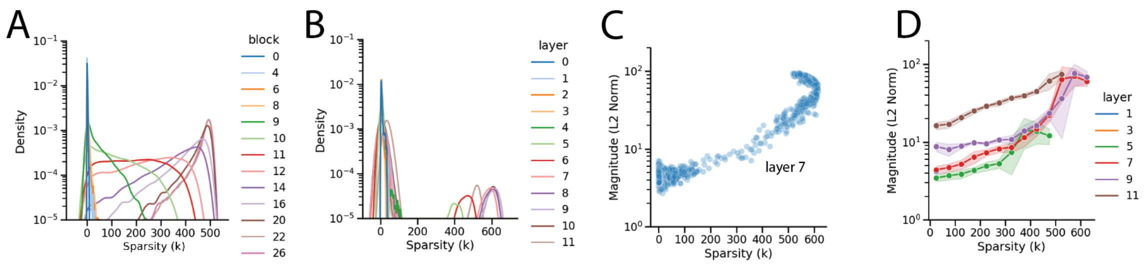
Figure 6: Layer-specific sparsity distributions of feature vectors as well as a direct relation between sparsity and the magnitude of feature vectors. A-B. ConvNext (A) has a unimodal distribution of sparsity across feature vectors in each block, while ViT-CLiP (B) maintains a bimodal distribution within each individual layer once sparse codes emerge. Distributions are depicted with kernel density estimate plots. C. A direct relationship is observed between the magnitude and sparsity of feature vectors for a representative layer (layer 7). D. As in C, but across all layers, where the average and standard deviation is shown for each of the binned sparsity values (bin width = 50).

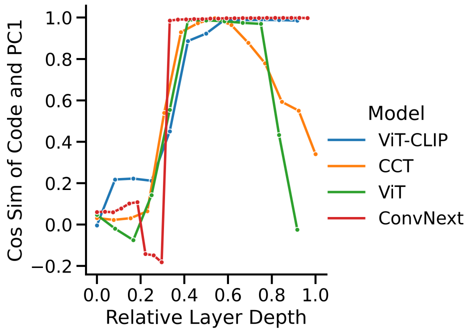
Figure 7: Sparse codes have high similarity to the first principal component of all feature vectors at layers after the sparse code emerges.

The high similarity between the sparse code and the first principal component demonstrates that this sparse code has a major influence on any analysis of the representations in these models, such as analyses of representational similarity across networks or between networks and brains (17).

3.4 Interpretation as a null code

Two main observed properties lead to the interpretation of the repeated sparse code as a "null code". First, by inspection, the locations of these emergent sparse feature vectors correspond to background, non-diagnostic image regions. This is observed for both transformer (Fig. 2) and convolutional (Fig. 3) models across layers and blocks once these repeated sparse codes emerge. Second, the ubiquity of these null codes in feature vectors across presentations and at different spatial locations demonstrates their independence from coding specific input features.

3.5 A computational role of null codes in transformers with LayerNorm

In addition to having properties that suggest a null code, we identify a computational mechanism with LayerNorm in transformer networks that demonstrates a way in which the presence of these null codes can act to form an implicit mask. In transformer networks, LayerNorm is the first operation before the MLP or MHSA in each layer. With LayerNorm, a large value in a subset of dimensions will effectively squash the values in remaining dimensions. While a large activation pattern in any dimension will induce competition and the modulation of other dimensions, there is a computational advantage of having a common sparse unique code (which utilizes a small subset of dimensions) to trigger the modulation of other dimensions. By utilizing a small subset of dimensions, there will be less interference with other input-related representations. An example of the null code being used as an implicit mask can be seen in Fig 8. Given the distributed tuning of network weights to the null code as outlined in A.2, however, it is likely that the computational mechanism of null codes as a direct implicit mask is only one way in which this sparse null code may be utilized.

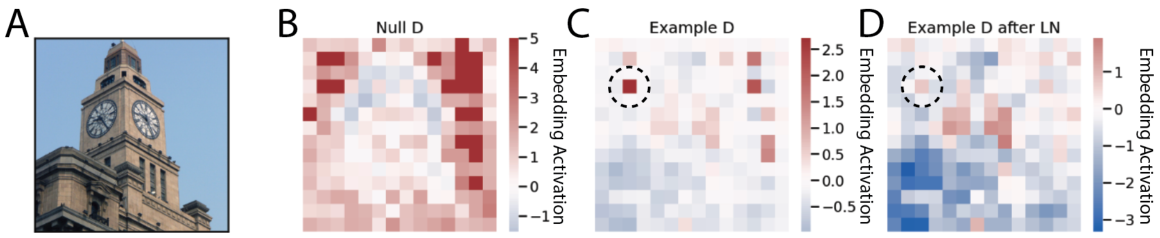

Figure 8: A computational mechanism with LayerNorm in Transformer networks demonstrates a way in which the presence of a null code can impose an implicit mask. A. Example image probe. B. In an example with a $D \times N$ embedding of layer 9 in ViT-CLIP, the distribution of the null code across the N token locations can be visualized as the activation level for the single dimension of D that has the highest magnitude in the null code. Note that the range in the visualization is clipped at a value of 5 for clarity. C. In the embeddings at a different dimension of D, certain token locations have high activations at the same locations as where the null code is present (circled). D. For the same dimension as depicted in C, the relative activation levels across all token locations is modified after applying LayerNom. Token locations where the null code is also present at high magnitudes (circled) are effectively squashed because LayerNorm is applied independently for each feature vector. The converse effect also occurs, where token locations that do correspond to salient image features locations (without the presence of a null code) have higher relative activation levels after LayerNorm.

4 Discussion

The emergence of sparse null codes in deep neural networks has implications in network design, interpretability, and enhancing a fundamental understanding of these systems. One of the most direct ways in which these sparse null codes may influence network design is through informing the development of normalization techniques (2). For example, while modifying LayerNorm to GroupNorm has been investigated in relation to stabilizing training and informative propagation (23), the interaction of LayerNorm with a subset of null dimensions suggests an additional theoretical tool that can be used to augment or regularize normalization. Furthermore, the location and prevalence of LayerNorm has shown to be critical in performant networks ((32; 22), which could be further investigated in relation to these sparse null codes. For informing network design, quantifying sparse null codes may be a useful tool that can be investigated more generally in relation to performance. Analysis of the location of sparse null codes may also be used as a tool to augment existing interpretability methods (25; 3; 5; 1) to characterize which input features are being utilized in computation.

There are several ways in which these sparse null codes may contribute to our fundamental understanding of modern neural networks. While there is a wide field of research investigating sparsity in neural networks, our work is most related to sparsity in network activation (19; 20). Typically, however, research on investigating sparse network activation is focused on absolute sparsity by focusing on areas of the network which have ReLU units (20) or impose some type of k-winner-take-all nonlinearity (24), rather than evaluating relative levels of sparsity such as with a measure of kurtosis. Additionally, our observation that there is a single highly sparse code is a qualitatively different phenomena than the general investigation of sparse codes in relation to topics such as compressed sensing (8), dictionary learning (10), and superposition (11).

These sparse null codes have implications for the analysis and interpretation of neural network representations. For example, any downstream analyses that rely on the geometrical properties of neural network representations such as brain representation similarity analysis (33; 16; 6; 17), will be influenced by these repeated sparse codes. Furthermore, the relevance of these sparse null codes to the overall neural network representations is substantial as the sparse null code approximates the first principal component of network representations in layers where the null code is present across all analyzed models.

The sparse codes observed in this work appear to be related to what have been called "outliers" in recent analyses of other transformer-based models. The most related observation is in concurrent work which identifies high-norm tokens in low-informative background areas of images in vision transformer based models (7). These observations (which identify outliers in DeIT, OpenCLIP, and DINOv2 models) are complementary to our findings and provide additional support for the generality of the trends observed here. Our work further establishes that outlier tokens in background areas are not only characterized by high norms but also by a single, shared sparse code. Our report also demonstrates that these sparse null codes can emerge in convolutional architectures and are not specific to transformers.

Other recent findings focus on outlier dimensions in transformer-based language models (12; 31; 18; 28; 30). The potential relationship between the sparse null codes observed in our work and the outlier dimensions observed in transformer-based language models remains an open question for future work, but, nonetheless, evidence for outlier dimensions in language models suggests the possibility that sparse null codes may be a more general phenomenon of neural networks that extends beyond vision models.

In this work, we identified a mechanism for how these sparse null codes can interact with LayerNorm in transformer networks to suppress feature activation in non-salient image locations. This LayerNorm mechanism may be complementary to additional ways in which the sparse null code can influence network activity. Supporting evidence of this view includes our observation that the query and key weights in the ViT-CLIP model had higher absolute levels of tuning to the null code, but with a mix of positive and negative values across a range of output dimensions, suggesting a distributed role of the null codes in the overall calculation of attention.

Perhaps the most surprising aspect of our results is the starkness with which this sparse null code emerges. In all of the models we have investigated, for every presentation, a highly consistent sparse code emerges at multiple layers. Our analyses show that these sparse null codes are present in some of the most popular models used by researchers. Interestingly, a sparse null code emerges even though there are fundamental differences in architectural backbone (convolutional and transformer) as well as training procedure (supervised vs. contrastive language image training). These sparse null codes also suggest that there may be implicit forms of attention calculation in neural networks (as opposed to explicit multi-headed self-attention). In sum, we identify and characterize a new class of highly abstract representations in deep vision models that have implications for network design, interpretability, and for theories of how these systems operate.

Acknowledgments and Disclosure of Funding

The work was supported by funding from an internal grant from the Johns Hopkins University Applied Physics Laboratory.

References

[1] Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. *arXiv preprint arXiv:2005.00928*, 2020.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 117(48):30071–30078, 2020.

[4] Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. Activation atlas. *Distill*, 4(3):e15, 2019.

[5] Hila Chefer, Shir Gur, and Lior Wolf. Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 782–791, June 2021.

[6] Colin Conwell, Jacob S Prince, George Alvarez, and Talia Konkle. Large-scale benchmarking of diverse artificial vision models in prediction of 7t human neuroimaging data. *bioRxiv*, 2022.

[7] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. *arXiv preprint arXiv:2309.16588*, 2023.

[8] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[9] A Dosovitskiy, L Beyer, A Kolesnikov, and others. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv*, 2020.

[10] Michael Elad. *Sparse and redundant representations: from theory to applications in signal and image processing*, volume 2. Springer, 2010.

[11] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

[12] Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. Representation degeneration problem in training natural language generation models. *arXiv preprint arXiv:1907.12009*, 2019.

[13] Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. Escaping the big data paradigm with compact transformers. *arXiv preprint arXiv:2104.05704*, 2021.
[14] 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.
[15] Niall Hurley and Scott Rickard. Comparing measures of sparsity. *IEEE Transactions on Information Theory*, 55(10):4723–4741, 2009.
[16] Seyed-Mahdi Khaligh-Razavi and Nikolaus Kriegeskorte. Deep supervised, but not unsupervised, models may explain it cortical representation. *PLoS computational biology*, 10(11):e1003915, 2014.
[17] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR, 2019.
[18] Olga Kovaleva, Saurabh Kulshreshtha, Anna Rogers, and Anna Rumshisky. Bert busters: Outlier dimensions that disrupt transformers. *arXiv preprint arXiv:2105.06990*, 2021.
[19] Mark Kurtz, Justin Kopinsky, Rati Gelashvili, Alexander Matveev, John Carr, Michael Goin, William Leiserson, Sage Moore, Nir Shavit, and Dan Alistarh. Inducing and exploiting activation sparsity for fast inference on deep neural networks. In *International Conference on Machine Learning*, pages 5533–5543. PMLR, 2020.
[20] Zonglin Li, Chong You, Srinadh Bhojanapalli, Daliang Li, Ankit Singh Rawat, Sashank J Reddi, Ke Ye, Felix Chern, Felix Yu, Ruiqi Guo, et al. The lazy neuron phenomenon: On emergence of activation sparsity in transformers. In *The Eleventh International Conference on Learning Representations*, 2022.
[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.
[22] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.
[23] Ekdeep Singh Lubana, Robert P Dick, and Hidenori Tanaka. Beyond BatchNorm: Towards a unified understanding of normalization in deep learning. June 2021.
[24] Alireza Makhzani and Brendan Frey. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.
[25] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital signal processing*, 73:1–15, 2018.
[26] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. 2015.
[27] Bruno A Olshausen and David J Field. Sparse coding of sensory inputs. *Current opinion in neurobiology*, 14(4):481–487, 2004.
[28] Giovanni Puccetti, Anna Rogers, Aleksandr Drozd, and Felice Dell'Orletta. Outliers dimensions that disrupt transformers are driven by frequency. *arXiv preprint arXiv:2205.11380*, 2022.
[29] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. Feb. 2021.
[30] William Rudman and Carsten Eickhoff. Stable anisotropic regularization. *arXiv preprint arXiv:2305.19358*, 2023.
[31] William Timkey and Marten van Schijndel. All bark and no bite: Rogue dimensions in transformer language models obscure representational quality. *arXiv preprint arXiv:2109.04404*, 2021.
[32] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep transformer models for machine translation. *arXiv preprint arXiv:1906.01787*, 2019.
[33] Daniel LK Yamins, Ha Hong, Charles F Cadieu, Ethan A Solomon, Darren Seibert, and James J DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the national academy of sciences*, 111(23):8619–8624, 2014.
[34] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

A Appendix

A.1 Sparse Repeated Code Characterization in ViT and CCT Transfomer Networks

In ViT and CCT transformer models, similar trends are observed (Fig A.1) as with ViT-CLIP with a bimodal distribution of sparsity in feature vectors, high similarity between the sparsest codes, and all of the highly sparse feature vectors having high similarity with the sparse code.

A.2 Characterizing how sparse repeated codes emerge and are utilized

Measuring input and output weight tuning to the sparse code in Transformers To investigate how these sparse codes emerge and are utilized, we characterize how these sparse codes relate to the

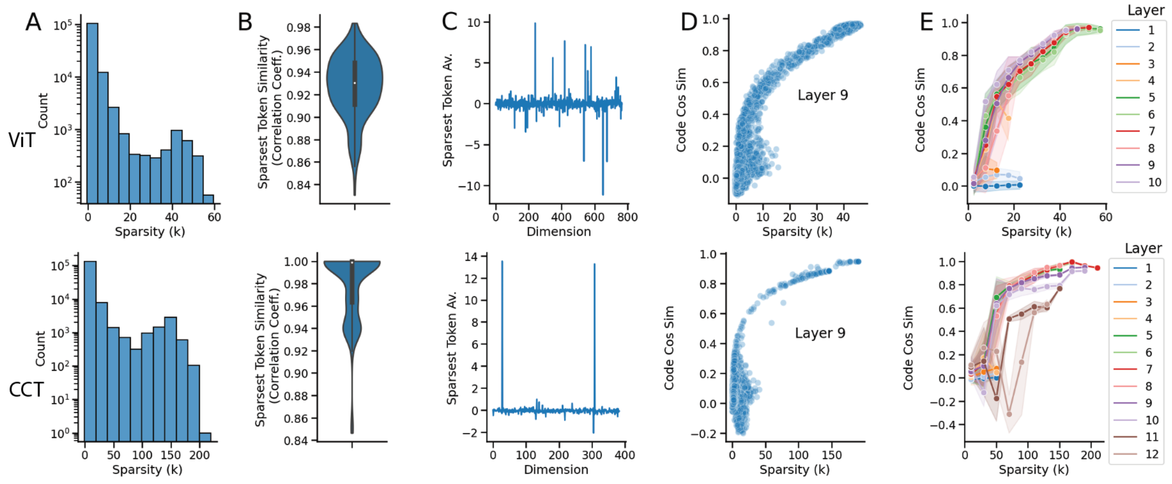

Figure A.1: In ViT and CCT Transformer model architectures, similar trends are observed as with ViT-CLIP with a bimodal distribution of sparsity in feature vectors, high similarity between the sparsest codes, and all of the highly sparse feature vectors having high similarity with the sparse code. A-E. As in Fig 3. with 24 input images.

input and output network weights in across the tranformer architecture. In transformer networks, each layer contains a multi-headed self-attention (MHSA) component and a multi-layer-perceptron (MLP) component. Given a shared embedding size of $N \times D$, the input weights for each of these components is size $D \times D^{in}$ and the output weights for each of these components is $D^{out} \times D$. Thus, the input and output network weights of each component are composed of D^{in} and D^{out} weight vectors which define the input and output linear projections of each component respectively. The D^{in} of the MHSA can be further sub-divided into the *query*, *key*, and *value* weights which perform distinct roles in processing input into the MHSA. To quantify the relation between the sparse code and these input or output weight vectors, we calculate the cosine similairity between the sparse code and all input or output weight vectors in the network (referred to as s^{in} and s_{out}), which is a measure of the tuning of these weight vectors to the sparse code. Note that s^{in} and s_{out} are vectors with D^{in} and D^{out} elements respectively.

Sparse code weight tuning in ViT-CLIP An example of this sparse code weight tuning characterization can be found in Fig A.2, which demonstrates differences in trends between how these networks weights are input and output in the MHSA and MLP for a specific network layer. We summarize the weight tuning properties at each layer by taking the overall magnitude ($||s^{in}||$ and $||s^{out}||$) and evaluating the number of times s^{in} or s^{out} exceeds a threshold as compared to 5 different shuffled versions of the null code (Fig A.2).

We identify properties in this tuning that is consistent between layers (Fig A.3). The *query* and *key* MHSA input weight vectors are consistently more tuned to the sparse codes than the *value* weight vectors. Additionally, the MLP input weight vectors are consistently less tuned to the sparse code at earlier layers of the network. The MLP has the unique properties of having a small number of output weight vectors that are highly tuned to the null code across layers (Fig A.3 E), but both the MLP and MHSA have output weights that are consistently tuned to the null code over several layers in the network (Fig A.3C-D).

Measuring input and output weight tuning to the sparse code tuning in ConvNext For analyzing how the sparse code is input and output across ConvNext blocks (Fig A.4), we can leverage the observation that ConvNext's sparse code is mostly confined to a single channel, and will refer to this channel as the "null" channel for subsequent analysis. Instead of measuring weight vector tuning by calculating the cosine similarity of the sparse code to each weight vector, we can more simply compare the null channel weight in each weight vector to the weights in the remaining channels. Note that taking the value of a single null channel in the input or output weights has D^{in} or D^{out} elements respectively, comparable to the cosine similarity measurement used for the Transformer architecture. In order to exclude channels that may be slightly tuned to the null code in our comparisons, we compare the null channel to the 90% of channels that have the smallest magnitude in the ConvNext's

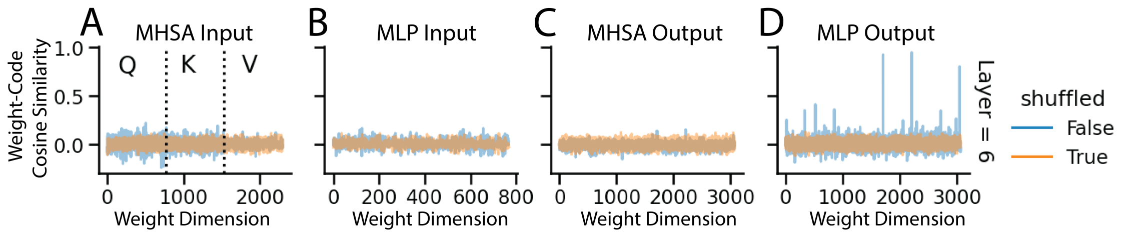

Figure A.2: Differences in the tuning of the MHSA and MLP input and output weight vectors to the sparse code is observed. A-D. The cosine similarity is measured between the sparse code and each input and output weight vector of the MHSA and MLP components. Each weight dimension across the x-axis corresponds to a unique weight vector that is measured in the MLP and the MHSA. Q,K, and V labels correspond to query, key, and value MHSA input weights respectively. A shuffled version of the sparse code is used for comparison.

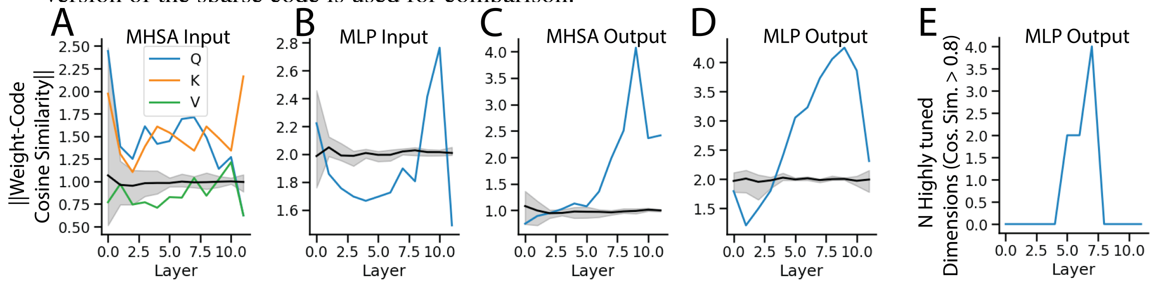

Figure A.3: Across layers, consistent differences in the tuning of the MHSA and MLP input and output weights to the sparse code is observed. A-D. The magnitude of weight tuning is measured for each layer (a summary of tuning measurements depicted for a single layer in Fig A.2). Black error bars represent maximum and minimum values measured for five random shuffles of the sparse code. E. The MLP has the unique properties of having a small number of output weight vectors that are highly tuned to the null code (not observed in the MHSA or any of the shuffles of the sparse code).

average sparse code. Due to architectural differences, inputs to each ConvNext block utilize a single depthwise convolutional filter with shape $D \times 7 \times 7$, which can be related to our previous analysis of having D^{in} =49 total weight vectors (7×7) linearly projecting feature vectors as input to each block.

Sparse code weight tuning in ConvNext Unlike the collection of weight vectors that linearly project feature vectors as input in Transformer components, the weight vectors in each convolutional filter have a spatial correspondence. A notable trend that we observe is that in the latest blocks (last 3 out of 27), the input convolutional filter weights become sharply tuned in the null channel (Fig A.4 A). The embeddings after applying the convolutional filter in a block is still shape $N \times D$, with N total feature vectors and we can compare the magnitude of the null channel activation to the remaining channel activations to quantify how the null code is input in each block for subsequent computation. We find that in the final blocks that have the sharper convolutional features, the null channel activation is higher than any of the other channels (Fig A.4B). An interpretation of this observation is that while the sparse codes emerge around block 9 out of 27, it is not until the final few blocks that the null code dominates the embeddings that after the application of the convolutional filters. For quantifying how these null codes relate to ConvNext block output weights, similar to the consistent null code output tuning of ViT-CLIP across layers once the null code emerges, the MLP in ConvNext has weights that are consistently tuned to the null channel (as opposed to remaining channels) across blocks (Fig A.4C). Note though, unlike the MLP in ViT-CLIP, we do not observe a small number of output weight vectors that are highly tuned to the null code.

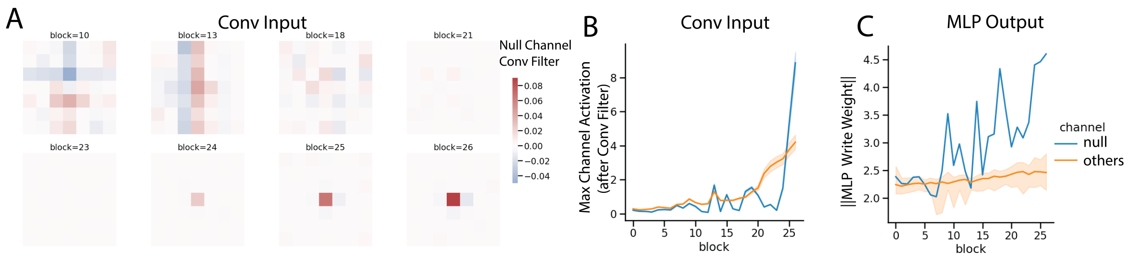

Figure A.4: Sparse code input and output tuning in ConvNext. A. Across blocks, the 7×7 slices of the convolutional filters corresponding to the null channel display spatial sharpness focused on the center location in the final blocks. B. In the network embeddings as a response to an input probe after the convolutional filters, null channel activation is higher than any of the other channels at the final blocks. C. The output weights for each ConvNext block are consistently tuned to the null channel (as opposed to remaining channels) across blocks as measured by relative magnitude of weights between channels.