

UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

Foundations of Neuron-level Interpretability: Automation, Evaluation and Training

Permalink

<https://escholarship.org/uc/item/3v5085kq>

ISBN

9798193222682

Author

Oikarinen, Tuomas

Publication Date

2026-10-02

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA SAN DIEGO

Foundations of Neuron-level Interpretability: Automation, Evaluation and Training

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Tuomas Oikarinen

Committee in charge:

Professor Lily Weng, Chair
Professor Julian McAuley
Professor Hao Su
Professor Berk Ustun

Copyright

Tuomas Oikarinen, 2026

All rights reserved.

The Dissertation of Tuomas Oikarinen is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2026

TABLE OF CONTENTS

Dissertation Approval Page	iii
Table of Contents	iv
List of Figures	ix
List of Tables	xiv
Acknowledgements	xviii
Vita	xx
Abstract of the Dissertation	xxii
Introduction	1
Chapter 1 CLIP-Dissect: Automatic Description of Neuron Representations in Deep Vision Networks	4
1.1 Introduction	4
1.2 Background and Related Work	5
1.3 Method	8
1.3.1 CLIP-Dissect Overview	9
1.3.2 Similarity function	10
1.3.3 Compatibility with future models	12
1.4 Experiments	12
1.4.1 Qualitative results	13
1.4.2 Quantitative results	14
1.4.3 Choice of similarity function	16
1.4.4 Computational efficiency	16
1.4.5 Detecting concepts missing from probing dataset	17
1.5 Use case of CLIP-Dissect	18
1.6 Limitations and conclusions	19
1.7 Acknowledgements	20
Chapter 2 Linear Explanations for Individual Neurons	22
2.1 Introduction	22
2.2 Motivation: How Important are Different Parts of a Neuron's Activation Pattern?	23
2.2.1 Definitions	24
2.2.2 Results	25
2.3 Method	26
2.3.1 Constructing a concept activation matrix	27
2.3.2 Learning Linear Explanations	28
2.4 Improving Evaluation of Explanations via Simulation	30

2.5	Experiment Results	35
2.5.1	Qualitative results	36
2.5.2	Simulation results: Correlation Scoring	37
2.5.3	Simulation results: Ablation scoring	38
2.6	Related Work	39
2.6.1	Automated Interpretability in Vision	39
2.6.2	Language Models Can Explain Neurons in LMs	40
2.6.3	Unreliability of explaining only top activations	41
2.6.4	Linear Probes	41
2.7	Conclusion	41
Chapter 3	Evaluating Neuron Explanations: A Unified Framework with Sanity Checks	43
3.1	Introduction	43
3.2	Definitions	44
3.2.1	What is an individual unit in a neural network?	44
3.2.2	Problem Description	45
3.2.3	What are the goals of an explanation?	46
3.3	A Unified Evaluation Framework	46
3.3.1	Formalizing Example Evaluations	47
3.3.2	Unifying Neuron Explanation Evaluation: <i>NeuronEval</i>	50
3.4	Meta-Evaluation #1: Sanity Checks for Evaluation Metrics	52
3.4.1	A Motivating Example	52
3.4.2	Sanity Test Definitions	54
3.4.3	Test Setup	56
3.4.4	Results and Discussion	57
3.5	Meta-Evaluation #2: Performance on Neurons with Known Concepts	59
3.5.1	Method: AUPRC between scores of correct and incorrect explanations	59
3.6	Discussion	61
3.6.1	Relevance of our tests to Realistic Failure modes	61
3.6.2	Conclusions	62
Chapter 4	Beyond Top Activations: Efficient and Reliable Crowdsourced Evaluation of Automated Interpretability	64
4.1	Introduction	64
4.2	Related Work	66
4.2.1	Automated Interpretability for Vision	66
4.2.2	Human Eval of Automated Interpretability	68
4.3	Methods	69
4.3.1	Model-Guided Importance Sampling (MG-IS)	70
4.3.2	Bayes Rating Aggregation (BRagg)	74
4.4	Experimental Validation and Human Study Design: Combining MG-IS and BRagg	77
4.5	Large-Scale Crowdsourced Study	79
4.5.1	Method Selection via Automated Evaluation	80

4.5.2	Crowdsourced Evaluation	81
4.6	Conclusion	82
Chapter 5	Label-Free Concept Bottleneck Models	84
5.1	Introduction	84
5.2	Related Work	85
5.3	Label-free CBM: A new framework to build CBM	88
5.3.1	Step 1: Concept set creation and filtering	88
5.3.2	Steps 2 and 3: Learning the Concept Bottleneck Layer (CBL)	90
5.3.3	Step 4: Learning the sparse final layer	91
5.4	Experiment Results	92
5.5	Case study: Manually improving an ImageNet model	96
5.5.1	Types of model errors	96
5.5.2	Editing final layer weights	98
5.6	Conclusion	99
Appendix A	Supplementary Material for Chapter 1	101
A.1	Additional Results and Details	101
A.1.1	Similarity function details and derivation	101
A.1.2	Additional qualitative results	104
A.1.3	Low level concepts	108
A.1.4	Compositional Concepts	111
A.1.5	Vision Transformer	111
A.1.6	Predicting input class from neuron descriptions	113
A.1.7	Visualizing wider range of activations	114
A.1.8	Interpretability cutoff	117
A.1.9	Qualitative effect of similarity function	117
A.1.10	Larger scale experiment on description quality	118
A.2	Crowdsourced evaluation of description quality	127
A.2.1	Setup	127
A.2.2	Results	128
Appendix B	Supplementary Material for Chapter 2	130
B.1	Appendix: Additional Information	130
B.1.1	Limitations	130
B.1.2	Additional Related Work: Input Importance	131
B.1.3	Greedy search algorithm	131
B.1.4	Explanation length	132
B.2	Appendix: Ablations	134
B.2.1	Different simulator model	134
B.2.2	Different concept sets	134
B.2.3	Different explainer model	135
B.2.4	No greedy search	135
B.2.5	Scaling for Simulation with Ablation Scoring	136

B.3	Appendix: Additional Results	138
B.3.1	Additional layers.	138
B.3.2	Dead neurons in ViT	140
B.3.3	Additional Qualitative Results	142
Appendix C	Supplementary Material for Chapter 3	148
C.1	Metric Definition & Details	148
C.1.1	Binarization	148
C.1.2	Simulation vs Classification Framing	149
C.1.3	Additional Metric Definitions	151
C.1.4	Equivalences	153
C.2	Discussion	156
C.2.1	Generative Model based Evaluation and missing labels test	156
C.2.2	Limitations	157
C.2.3	Motivating Example for Sanity Checks	160
C.2.4	Additional Related Works	162
C.3	Theoretical Missing/Extra Labels Test and Concept Imbalance	165
C.3.1	Results	166
C.4	Analytical solution for Missing/Extra Labels Test	173
C.4.1	Special case - Theoretical Test	176
C.5	Ablations and Extra checks	177
C.5.1	Increase/decrease Fraction in Missing/Extra labels test	177
C.5.2	Failure case of cosine: Adding a constant activation	179
C.5.3	Random vs Semantic subsets	181
C.5.4	Epsilon ablation	182
C.5.5	Number of samples ablation	184
C.5.6	Combination Metrics	185
C.6	Detailed results	188
C.6.1	Missing and Extra Labels Test:	188
C.6.2	Meta-AUPRC evaluation:	192
Appendix D	Supplementary Material for Chapter 4	196
D.1	Discussion	196
D.1.1	Limitations	196
D.1.2	Broader Impact	197
D.1.3	Crowdsourced Study Design	197
D.2	Ablation Studies	198
D.2.1	Choice of γ	198
D.2.2	Sensitivity to β in BRagg (Uniform Prior)	199
D.2.3	Activation-Guided Importance Sampling	201
D.2.4	Error Model for Rating Aggregation	202
D.2.5	SigLIP Only Evaluation Accuracy	203
D.2.6	Compositional/Complex Explanations	204
D.2.7	Model Ablation	205

D.2.8	Effect of Sampling Model Accuracy on MG-IS Performance	208
D.2.9	Comparison with Downstream Task Evaluation	208
D.3	Additional Details	210
D.3.1	MTurk Experimental Details:	210
D.3.2	Detailed MTurk results	211
D.3.3	Experimental Details	216
D.3.4	Automated Evaluation Details	218
D.3.5	Theorem 1	219
D.3.6	Compute details	221
D.3.7	Inter-Annotator Consistency	221
Appendix E	Supplementary Material for Chapter 5	223
E.1	Additional Details and Results	223
E.1.1	Appendix Overview	223
E.1.2	Limitations	224
E.1.3	Dissection performance of cos cubed similarity function	224
E.1.4	Ablation: Effect of concept filters	226
E.1.5	Further discussion and example of concept filtering	227
E.1.6	Ablation: Effect of Initial Concept set	229
E.1.7	Ablation: LF-CBM without sparsity constraint	232
E.1.8	Consistency of concept set generation	233
E.1.9	More details on Model editing procedure	235
E.1.10	Additional figures	237
E.1.11	Additional weight visualizations for random classes	241
E.1.12	Additional explanations for random images	244
E.2	User Study on the Interpretability of LF-CBM	248
E.2.1	Task 1: Do neurons in CBL correspond to their target concept?	248
E.2.2	Task 2: How reasonable are our explanations for a single decision?	250
Bibliography		254

LIST OF FIGURES

Figure 1.	Overview of the 3 main directions of the dissertation: Automation, Evaluation and Training in Neuron-level Interpretability.	1
Figure 1.1.	Labels generated by our method CLIP-Dissect, NetDissect [6] and MILAN [39] for random neurons of ResNet-50 trained on ImageNet.	6
Figure 1.2.	Overview of CLIP-Dissect: a 3-step algorithm to dissect neural network of interest	9
Figure 1.3.	Example of a final layer neuron: we compare the descriptions generated by different methods and our metrics.	16
Figure 1.4.	Example of CLIP-Dissect correctly labeling neurons that detect the little blue heron and the great white heron without using any bird images.	18
Figure 1.5.	Showcasing similarity between neurons connected by high weights.	19
Figure 2.1.	Overview of our proposed method: Linear Explanations.	25
Figure 2.2.	An overview of the simulation pipeline with correlation scoring.	30
Figure 2.3.	An area chart of the activations of two neurons in layer4 of ResNet-50.	33
Figure 2.4.	Descriptions and highly activating images from different ranges of example neurons.	35
Figure 2.5.	The relationship between correlation and ablation score of different explanations.	39
Figure 3.1.	An overview of NeuronEval contributions.	45
Figure 3.2.	Overview of our theoretical Missing and Extra Labels Tests.	53
Figure 3.3.	The effect of Concept/Neuron Activation frequency on the score difference in Theoretical Extra Labels test.	58
Figure 4.1.	Overview of the explanation evaluation pipeline.	65
Figure 4.2.	Comparing different sampling strategies to estimate ρ_S	73
Figure 4.3.	Comparison of different rating aggregation strategies on simulated human study with error rate $\eta = 23\%$, tested on the full dataset.	75
Figure 4.4.	Comparing the effect of our sampling technique MG-IS, rating aggregation method BRAGg independently and both together.	77

Figure 4.5.	Results of our MTurk study	80
Figure 5.1.	Our proposed Label-free CBM has many desired features which existing CBMs lack.	84
Figure 5.2.	Overview of our pipeline for creating Label-free CBM.	87
Figure 5.3.	Visualization of the final layer weights of our Label-free CBM. Showcasing how our models differentiate between two similar classes.	95
Figure 5.4.	Visualization of two correct decisions made by our Label-free CBM.	96
Figure 5.5.	Fixing an incorrect model prediction.	97
Figure 5.6.	Examples of two edits we performed on our model predictions.	98
Figure A.1.	Explanations of most interpretable neurons in the second to last layer of ResNet-18 trained on Places365.	106
Figure A.2.	Explanations of most interpretable neurons in the second to last layer of ResNet-50 trained on ImageNet	107
Figure A.3.	Descriptions for select neurons in early layers of ResNet-152.	109
Figure A.4.	Descriptions of most interpretable neurons (highest similarity/IoU) of an early layer in ResNet-50.	110
Figure A.5.	An example of compositional explanations generated by our method for two neurons of ResNet50 trained on ImageNet.	111
Figure A.6.	Most interpretable neurons after the 'encoder' module (one of the last layers) of a Vision Transformer.	112
Figure A.7.	Most interpretable neurons of ResNet-18, showcasing randomly sampled images of a wide range of most highly activating images for that neuron.	115
Figure A.8.	Most interpretable neurons of ResNet-50, showcasing randomly sampled images of a wide range of most highly activating images for that neuron.	116
Figure A.9.	Qualitative difference between similarity functions.	118
Figure A.10.	Example of the user interface for our evaluation of description quality.	119
Figure A.11.	Random neurons of 'conv1' in ResNet-50 (ImageNet), and E1, E2's evaluation on the description of CLIP-Dissect (PART 1).	121

Figure A.12.	Random neurons of 'conv1' in ResNet-50 (ImageNet), and E1, E2's evaluation on the description of CLIP-Dissect (PART 2)	122
Figure A.13.	Random neurons of 'conv1' in ResNet-50 (ImageNet), and E1, E2's evaluation on the description of CLIP-Dissect (PART 3)	123
Figure A.14.	Random neurons of 'layer 4' in ResNet-18 (Places 365), and E1, E2's evaluation on the description of CLIP-Dissect (PART 1)	124
Figure A.15.	Random neurons of 'layer 4' in ResNet-18 (Places 365), and E1, E2's evaluation on the description of CLIP-Dissect (PART 2)	125
Figure A.16.	Random neurons of 'layer 4' in ResNet-18 (Places 365), and E1, E2's evaluation on the description of CLIP-Dissect (PART 3)	126
Figure A.17.	User Interface for our Mechanical Turk study.	128
Figure B.1.	Example interpretable neurons.	142
Figure B.2.	Example interpretable neurons.	143
Figure B.3.	Randomly selected neurons.	143
Figure B.4.	Randomly selected neurons.	144
Figure B.5.	Randomly selected neurons.	144
Figure B.6.	Example interpretable neurons of ResNet-18(Places365).	145
Figure B.7.	Example interpretable neurons of ResNet-18(Places365).	145
Figure B.8.	Example interpretable neurons of VGG-16(CIFAR-100).	146
Figure B.9.	Example interpretable neurons of ViT-B/16(ImageNet).	146
Figure B.10.	Example interpretable neurons of ViT-L/32.	147
Figure C.1.	Motivating example: A hypothetical neuron that only activates on pets (dogs or cats).	161
Figure D.1.	Comparing rating aggregation strategies in Setting 2 (Testing on MTurk) described in Sec 4.4.	198
Figure D.2.	The Effect of the γ hyperparameter on Importance sampling performance.	199
Figure D.3.	Comparing the error rate of using different β values for BRagg(Uniform prior).	199

Figure D.4.	Comparing different sampling strategies on simulated data with no label noise.	201
Figure D.5.	Comparing different error models for rating aggregation. Note log-scale on both axes.	203
Figure D.6.	Comparison the error of different rating aggregation strategies and only relying on SigLIP instead of conducting a human study.	204
Figure D.7.	Model ablation for guided sampling.	206
Figure D.8.	Comparing different choices for the Bayes prior. We can see SigLIP2 performs the best overall.	207
Figure D.9.	Our user study interface.	210
Figure D.10.	Visualization of example neurons, their descriptions and correlations scores from our crowdsourced evaluation.	213
Figure D.11.	Visualization of example neurons, their descriptions and correlations scores from our crowdsourced evaluation.	214
Figure D.12.	Visualization of example neurons, their descriptions and correlations scores from our crowdsourced evaluation.	215
Figure E.1.	<i>Comparison of the final layer weights between CIFAR-10 model trained with and without filters.</i>	230
Figure E.2.	Dense CBM explanation for the same image as the fourth image in Figure E.11. We can see the dense model is practically uninterpretable.	232
Figure E.3.	Difference in concepts used for a decision and their weights for models trained with original vs regenerated concept set.	234
Figure E.4.	Full prompts used for GPT-3 concept set creation. Text in green generated by GPT, rest is our prompt.	237
Figure E.5.	Full prompts used for GPT-3 concept set creation. Text in green generated by GPT, rest is our prompt.	238
Figure E.6.	The other 3 model edits we performed in our experiment	239
Figure E.7.	Examples of Type 1 and type 2 model errors.	240
Figure E.8.	Weight visualizations for 4 randomly chosen output classes for our CBM trained on Places365.	242

Figure E.9.	Weight visualizations for 4 randomly chosen output classes for our CBM trained on ImageNet.	243
Figure E.10.	Explanations for 4 randomly chosen input images for our CBM trained on Places365.....	245
Figure E.11.	Explanations for 4 randomly chosen input images for our CBM trained on ImageNet.	246
Figure E.12.	Explanations for 4 randomly chosen input images for our CBM trained on CUB-200.	247
Figure E.13.	User Interface shown to Mechanical Turk users for Task 1.	249
Figure E.14.	User Interface shown to Mechanical Turk users for Task 2. In this example Model 1 is LF-CBM while Model 2 is standard dense model.	251
Figure E.15.	Crowdsourced results for Task 2, without filtering any responses.	253
Figure E.16.	Crowdsourced results for Task 2, where we filter out responses from any worker who gave at least one inconsistent response.	253

LIST OF TABLES

Table 1.1.	The cosine similarity of predicted labels compared to ground truth labels on final layer neurons of ResNet-50 trained on ImageNet.	14
Table 1.2.	Performance when labeling final layer neurons of a ResNet18 trained on Places365.	15
Table 1.3.	Comparison of the performance between similarity functions.	17
Table 1.4.	Runtime comparison.	17
Table 2.1.	Average <i>Top Impact</i> for neurons in different layers of ResNet-50 (ImageNet).	26
Table 2.2.	Average correlation scores between simulated and actual neuron activations.	37
Table 2.3.	Average Ablation scores between simulated and actual neuron activations, average over all neurons in layer4.	37
Table 3.1.	Summary of evaluations used in existing work.	49
Table 3.2.	A 'pet' neuron: different metrics M give different evaluation scores s_M on explanations t	53
Table 3.3.	Averaged experimental and theoretical results of our missing labels and extra labels test.	55
Table 3.4.	Comparison of different evaluation metrics on neurons with known concepts, averaged across 8 settings.	60
Table 4.1.	SigLIP based simulation with correlation scoring comparing different simple explanation methods.	80
Table 5.1.	Comparison of our method against existing methods for creating Concept Bottleneck models	87
Table 5.2.	Accuracy comparison of CBM methods.	94
Table A.1.	The average description scores of the CLIP-Dissect descriptions for neurons of different layers.	120
Table A.2.	Results of our crowdsourced experiment.	129
Table B.1.	Average Explanation lengths of our method for second to last layer neurons of different models discussed in Table 2.2.	132
Table B.2.	Correlation scores of different explanations in layer4 of the models.	134

Table B.3.	Concept set effect on correlation scores.	135
Table B.4.	Comparison of different explainer models for LE(SigLIP) on laeyr4 neurons of ResNet-50(ImageNet).	135
Table B.5.	Average correlation score of different explanations for neurons in layer4 of ResNet-50(ImageNet). Number in brackets is the average description length.	136
Table B.6.	Testing the scaling function for ablation scoring with simulation.	137
Table B.7.	Average correlation scores of explanations by different methods on different layers of ResNet-50 trained on ImageNet.	138
Table B.8.	Average correlation scores of explanations by different methods on different layers of ResNet-18 trained on Places365.	138
Table B.9.	Average correlation score across neurons in different layers of VGG16. ...	139
Table B.10.	Average simulation correlation scores of different methods for different layers of ViT-B-16.	140
Table B.11.	Fraction of dead neurons in different models.	140
Table B.12.	Average correlations scores of different explanation methods without dead neurons on ViT.	141
Table C.1.	Definition of commonly-used binary classification metrics.	155
Table C.2.	Definition of other commonly-used metrics. Here, $x, y \in \mathbb{R}^N$ are two real vectors, $\bar{x}, \bar{y}$ refer to the mean of x, y	155
Table C.3.	Extended Table (of Table 3.1 in the main text section 3.4) comparing related work.	168
Table C.4.	Average Decrease Accuracy on Theoretical Missing labels Test. A metric fails if it has low decrease accuracy on any concept frequency γ	169
Table C.5.	Average Decrease Accuracy on Theoretical Extra labels Test. A metric fails if it has low decrease accuracy on any concept frequency γ	170
Table C.6.	Simulation results of missing labels test on idealized neuron with perfect correspondence to a concept activation.	171
Table C.7.	Simulation results of extra labels test on idealized neuron with perfect correspondence to a concept activation.	172

Table C.8.	The evaluation scores for different metrics under missing and extra label tests on an <i>ideal</i> neuron.	177
Table C.9.	Further simplifying from Table C.8 by plugging in p values we typically use in our tests.	177
Table C.10.	Ablation study on sanity checks varying the magnitude of change in labels.	179
Table C.11.	Adding a constant to the activation values of all neurons causes the cosine similarity to perform very poorly, while other metrics are unchanged.	180
Table C.12.	Comparison of c_t^- derived as a random subclass of c_t vs a real semantic subclass of c_t derived from the WordNet hierarchy.	181
Table C.13.	An Ablation study on the effect of ϵ choice on our results.	183
Table C.14.	A study measuring whether our test results are sensitive to sampling $c_t^\pm$ only once per neuron.	184
Table C.15.	Theoretical Missing Labels test on combination metrics.	185
Table C.16.	Theoretical Extra Labels Test for combination metrics.	186
Table C.17.	Experimental Missing and Extra Labels Test with new combination metrics.	186
Table C.18.	Performance of new combination on the meta-AUPRC evaluation introduced in section 3.5.	187
Table C.19.	Detailed results of Missing/Extra Labels test on Setting 1 and 2.	190
Table C.20.	Detailed results of Missing/Extra Labels test on Setting 3 and 4.	190
Table C.21.	Detailed results of Missing/Extra Labels test on Setting 5 and 6.	191
Table C.22.	Detailed results of Missing/Extra Labels test on Setting 7 and 8.	191
Table C.23.	Meta-AUPRC evaluation results on ImageNet Settings(1-4).	193
Table C.24.	Meta-AUPRC evaluation results on Places and CUB models Settings(5-8).	194
Table C.25.	Meta-AUPRC evaluation results on Language Models Settings(9-10).	195
Table D.1.	SigLIP based simulation with correlation scoring comparing complex explanations ($l > 1$).	205
Table D.2.	Large scale crowdsourced study results using SigLIP 2 as the prior for BRAgg.	207

Table D.3.	Comparing Sampling performance based on SigLIP AUC in detecting the concept. We report average RCE.	208
Table D.4.	Using generated explanations to find neurons in RN-50 that work as classifiers for unseen classes. We report Average AUC.	209
Table D.5.	Detailed results of our large scale crowdsourced study.	211
Table D.6.	Statistical significance of our results according to two-sample students t-test.	212
Table D.7.	Approximate runtime of different $l = 1$ baseline methods to explain neurons.	221
Table D.8.	Approximate runtime of different complex explanation baseline methods. .	221
Table E.1.	Comparison of the performance between similarity functions.	225
Table E.2.	Effect of our individual concept filters on the final accuracy and number of concepts used by our models.	227
Table E.3.	Accuracy comparison between using different methods for generating initial concept set.	231
Table E.4.	Results of training our CBM without sparsity constraints on the final layer.	232
Table E.5.	Mechanical Turk results of Task 1.	249

ACKNOWLEDGEMENTS

First, I would like to thank Professor Tsui-Wei (Lily) Weng for her invaluable guidance and support throughout the entire PhD as an advisor and committee chair. This dissertation would not be possible without her contributions and advice, including but not limited to multiple late nights revising paper drafts, helping choose the most impactful research directions and encouraging mathematical rigor and clarity of presentation throughout my research.

I would also like to thank my Committee of Professor Julian McAuley, Professor Hao Su and Professor Berk Ustun for their valuable feedback and advice.

Next, I would like to thank my labmates Chung-En Sun, Ge Yan and Akshay Kulkarni for providing great company as well as great collaboration through the years.

I would also like to thank all other amazing collaborators I have had the privilege to work with throughout the years including Subhro Das, Lam Nguyen, Luca Daniel, Wang Zhang, Alexandre Megretski, Max Langenkamp, Nicholas Bai, Rahul Iyer, Aya Ismail, Amy Wang, Julius Adebayo, Samuel Stanton, Kyunghyun Cho, Nathan Frey, Justin Lee, Arjun Chatha, Keng-Chi Chang, Yilan Chen, Sin-Han Yang, Anvita Srinivas, Divyansh Srivastava, Wei-Hung Weng, Mohammad Ali Khan, Tongqing Shi, Amirhosein Javadi, Chandan Singh, Charlotte Siska, Zixiao Chen, Jianfeng Gao, Yiwen Tu, Jerry Thomas John, You Chen and Bryan Maruyama.

Finally, I would like to thank my friends and family for all their support and keeping my spirits high regardless of how research was going. Special thanks to my parents for supporting me in pursuing my dreams even when it meant moving across the world, and Taylor for always being there for me despite many late nights of work and frequent travel to pursue my research.

Chapter 1, in full, is based on the paper from Tuomas Oikarinen, and Tsui-Wei Weng "CLIP-Dissect: Automatic Description of Neuron Representations in Deep Vision Networks" published at the International Conference on Learning Representations (ICLR), 2023. The dissertation author is the primary investigator and author of this paper.

Chapter 2, in full, is based on the paper from Tuomas Oikarinen and Tsui-Wei Weng, "Linear Explanations for Individual Neurons" published at the International Conference on

Machine Learning (ICML), 2024. The dissertation author is the primary investigator and author of this paper.

Chapter 3, in full, is based on the paper from Tuomas Oikarinen, Ge Yan and Tsui-Wei Weng, "Evaluating Neuron Explanations: A Unified Framework with Sanity Checks" published at the International Conference on Machine Learning (ICML), 2025. The dissertation author is the primary investigator and author of this paper.

Chapter 4, in full, is based on the paper from Tuomas Oikarinen, Ge Yan, Akshay Kulkarni and Tsui-Wei Weng, "Beyond Top Activations: Efficient and Reliable Crowdsourced Evaluation of Automated Interpretability" published at the Conference on Computer Vision and Pattern Recognition (CVPR), 2026. The dissertation author is the primary investigator and author of this paper.

Chapter 5, in full, is based on the paper from Tuomas Oikarinen, Subhro Das, Lam M Nguyen and Tsui-Wei Weng, "Label-free Concept Bottleneck Models" published at the International Conference on Learning Representations (ICLR), 2023. The dissertation author is the primary investigator and author of this paper.

VITA

2021	Bachelor of Science, Massachusetts Institute of Technology
2022	Research Intern, MIT-IBM Watson AI Labs
2024	Research Intern, Genentech - Prescient Design
2025	Research Intern, Microsoft Research
2021–2026	Research Assistant, University of California San Diego
2026	Doctor of Philosophy, University of California San Diego

PUBLICATIONS

”Robust Deep Reinforcement Learning through Adversarial Loss”, Tuomas Oikarinen, Wang Zhang, Alexandre Megretski, Luca Daniel and Tsui-Wei Weng, Neural Information Processing Systems (NeurIPS), 2021.

”CLIP-Dissect: Automatic Description of Neuron Representations in Deep Vision Networks”, Tuomas Oikarinen, and Tsui-Wei Weng, International Conference on Learning Representations (ICLR), 2023.

”Label-free Concept Bottleneck Models”, Tuomas Oikarinen, Subhro Das, Lam M Nguyen and Tsui-Wei Weng, International Conference on Learning Representations (ICLR), 2023.

”Concept Driven Continual Learning”, Sin-Han Yang, Tuomas Oikarinen and Tsui-Wei Weng, Transactions on Machine Learning Research, 2024.

”SAND: Enhancing Open-Set Neuron Descriptions through Spatial Awareness”, Anvita Srinivas, Tuomas Oikarinen, Divyansh Srivastava, Wei-Hung Weng and Tsui-Wei Weng, Winter Conference on Applications of Computer Vision (WACV), 2025.

”Linear Explanations for Individual Neurons”, Tuomas Oikarinen and Tsui-Wei Weng, International Conference on Machine Learning (ICML), 2024.

”Interpreting Neurons in Vision Networks with Language Models”, Nicholas Bai, Rahul Iyer, Tuomas Oikarinen and Tsui-Wei Weng, Transactions on Machine Learning Research, 2025.

”Evaluating Neuron Explanations: A Unified Framework with Sanity Checks”, Tuomas Oikarinen, Ge Yan and Tsui-Wei Weng, International Conference on Machine Learning (ICML), 2025.

”Concept Bottleneck Large Language Models”, Chung-En Sun, Tuomas Oikarinen, Berk Ustun and Tsui-Wei Weng, International Conference on Learning Representations (ICLR), 2025.

”Concept Bottleneck Language Models for Protein Design”, Aya Abdelsalam Ismail, Tuomas Oikarinen, Amy Wang, Julius Adebayo, Samuel Stanton, Taylor Joren, Joseph Kleinhertz, Allen Goodman, Hector Corrada Bravo, Kyunghyun Cho and Nathan C Frey, International Conference on Learning Representations (ICLR), 2025.

”Interpretable Generative Models through Post-hoc Concept Bottlenecks”, Akshay Kulkarni, Ge Yan, Chung-En Sun, Tuomas Oikarinen and Tsui-Wei Weng, Conference on Computer Vision and Pattern Recognition (CVPR), 2025.

”CI-CBM: Class-Incremental Concept Bottleneck Model for Interpretable Continual Learning”, Amirhosein Javadi, Tuomas Oikarinen, Tara Javidi and Tsui-Wei Weng, Transactions on Machine Learning Research, 2026.

”Beyond Top Activations: Efficient and Reliable Crowdsourced Evaluation of Automated Interpretability”, Tuomas Oikarinen, Ge Yan, Akshay Kulkarni and Tsui-Wei Weng, Conference on Computer Vision and Pattern Recognition (CVPR), 2026.

ABSTRACT OF THE DISSERTATION

Foundations of Neuron-level Interpretability: Automation, Evaluation and Training

by

Tuomas Oikarinen

Doctor of Philosophy in Computer Science

University of California San Diego, 2026

Professor Lily Weng, Chair

Despite the impressive performance of deep neural networks, our understanding of how they work is very limited. To address this, the field of mechanistic interpretability aims to build a bottom-up understanding of the mechanisms that a neural network uses to calculate its output. An important building block for mechanistic interpretability is interpreting the roles of individual neurons inside the network. However, interpreting individual neurons poses many challenges. First, a single network may have thousands or millions of neurons, so it is not feasible for humans to manually interpret each neuron. Second, the field lacks clear standardized definitions of what makes a good explanation, making it difficult to compare competing methods or measure progress. Finally, individual neurons can be polysemantic, i.e., they activate on

multiple unrelated concepts, which makes interpreting them difficult.

In this dissertation, I discuss solutions to these three essential problems in neuron-level interpretability. First, in Chapters 1 and 2, I introduce CLIP-Dissect and Linear Explanations, two methods that leverage pre-trained multimodal models to automatically explain all neurons of a vision model. Second, in Chapters 3 and 4, I discuss our two recent papers addressing the lack of standardized evaluation, where we unify diverse evaluations from existing studies under the same mathematical framework and find that most methods don't pass simple sanity checks, as well as propose ways to more efficiently evaluate these explanations. Finally, in Chapter 5, I discuss a solution to the challenge of polysemanticity in the form of Label-free Concept Bottleneck Models, an automated method for training models to have interpretable individual neurons.

Introduction

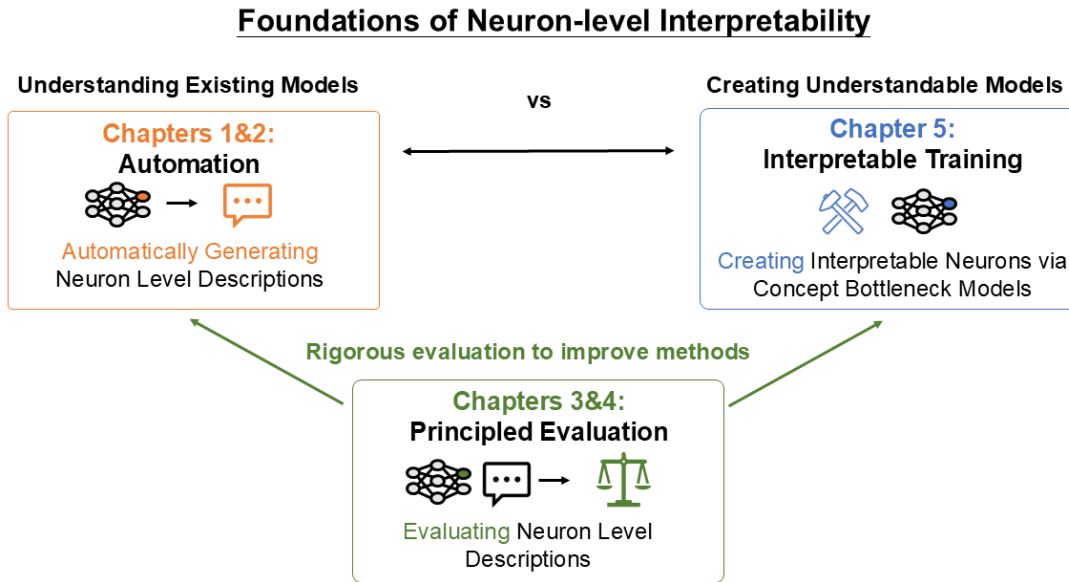


Figure 1. Overview of the 3 main directions of the dissertation: Automation, Evaluation and Training in Neuron-level Interpretability.

In recent years, neural networks have achieved very impressive performance across many domains like vision, language and coding, and are being increasingly relied on to make important decisions. However, our understanding of these models has not improved at a similar rate, making it hard to know why models are making certain decisions. This limits how much we can trust the models, and increases risks of models behaving in unexpected ways. To address this, the field of mechanistic interpretability has emerged to build a bottom-up, mechanistic understanding of the model’s function, with the ultimate goal of being able to reverse engineer the networks.

To build such a mechanistic understanding, it is important to first understand small units like individual neurons or directions in activation space inside the model. This is known as

neuron-level interpretability, and is the focus of this dissertation. However, understanding these individual neurons can be very challenging. First, a single network may have thousands or millions of neurons, so it is not feasible for humans to manually interpret each neuron. Instead automated methods are needed for neuron-level interpretability to be scalable and cost-efficient.

Second, the field lacks rigorous evaluation, with most papers relying heavily on qualitative evaluations and using different evaluation metrics without justification. This makes it difficult to compare competing methods or measure progress.

Finally, individual neurons can be complex and polysemantic, i.e., activate on multiple unrelated concepts which makes interpreting them difficult. This creates a need for more interpretable models or representations.

This dissertation focuses on addressing these three problems through Automation, Principled Evaluation and Interpretable Training as shown in Figure 1.

Chapters 1 and 2 discuss CLIP-Dissect [68] and Linear Explanations [69], two methods that leverage pre-trained multimodal models to automatically explain all neurons of a vision model. These automated interpretability methods address the first challenge and let us interpret a large number of neurons without human effort. In addition, Linear Explanations addresses the issue of polysemantic neurons by explaining a single neuron as a linear combination of interpretable concepts. Overall, we show these methods improve the faithfulness and efficiency of automated neuron explanations. See [4, 91, 55] for our other work in automated interpretability across vision and language domains.

Chapters 3 and 4 address the issue of lacking evaluation standards. In Chapter 3 we unify diverse evaluation methods from 20 existing studies under the shared NeuronEval [71] framework, and find that many studies use metrics that don't pass simple sanity checks. Based on these results, in Chapter 4 we propose a cost-efficient method to evaluate a reliable metric (correlation coefficient) via a human study [70], providing around a $40\times$ improvement in efficiency over standard methods.

Finally, in Chapter 5, I discuss an alternative solution to polysemantic neurons by training

the models to be more interpretable. In particular, I focus on Label-free Concept Bottleneck Models (LF-CBM) [67]. LF-CBM allows us to automatically transform an existing vision model to a more interpretable Concept Bottleneck Model, where each neuron of a layer corresponds to an interpretable concept. For additional information on creating interpretable models, see our work on Concept Bottleneck Language Models [95, 44], Concept-based Continual Learning [104, 45] or Concept Bottleneck generative vision models [53].

Overall, I believe addressing these three challenges lays the foundations of neuron-level interpretability, and allows us to work up towards a mechanistic understanding of the entire model.

Chapter 1

CLIP-Dissect: Automatic Description of Neuron Representations in Deep Vision Networks

1.1 Introduction

Deep neural networks (DNNs) have demonstrated unprecedented performance in various machine learning tasks spanning computer vision, natural language processing and application domains such as healthcare and autonomous driving. However, due to their complex structure, it has been challenging to understand why and how DNNs achieve such great success across numerous tasks. Understanding how the trained DNNs operate is essential to trust their deployment in safety-critical tasks and can help reveal important failure cases or biases of a given model.

One way towards understanding DNNs is to inspect the functionality of individual neurons, which is the focus of our work. This includes methods based on manual inspection [23, 110, 114, 73, 72, 31], which provide high quality explanations and understanding of the network but require large amounts of manual effort. To address this issue, researchers have developed automated methods to describe the functionality of individual neurons, such as Network Dissection [6] and Compositional Explanations [64]. In [6], the authors first created a new dataset named *Broden* with pixel labels associated with a pre-determined set of concepts, and then use *Broden* to find neurons whose activation pattern matches with that of a pre-defined

concept. In [64], the authors further extend Network Dissection to detect more complex concepts that are logical compositions of the concepts in *Broden*. Although these methods based on Network Dissection can provide accurate labels in some cases, they have a few major limitations: (1) They require a densely annotated dataset, which is expensive and requires significant amount of human labor to collect; (2) They can only detect concepts from the fixed concept set which may not cover the important concepts for some networks, and it is difficult to expand this concept set as each concept requires corresponding pixel-wise labeled data.

To address the above limitations, we propose CLIP-Dissect, a novel method to automatically dissect DNNs with *unrestricted* concepts *without* the need of any concept labeled data. Our method is training-free and leverages pre-trained multi-modal models such as CLIP [77] to efficiently identify the functionality of individual neuron units. We show that CLIP-Dissect (i) provides high quality descriptions for internal neurons, (ii) is more accurate at labeling final layer neurons where we know the ground truth, and (iii) is $10\times$ - $200\times$ more computationally efficient than existing methods. Finally, we show how one can use CLIP-Dissect to better understand neural networks and discover that neurons connected by a high weight usually represent similar concepts.

1.2 Background and Related Work

Network dissection.

Network dissection [6] is the first work on understanding DNNs by automatically inspecting the functionality (described as *concepts*) of each individual neuron¹. They identify concepts of intermediate neurons by matching the pattern of neuron activations to the patterns of pre-defined concept label masks. In order to define the ground-truth concept label mask, the authors build an auxiliary densely-labeled dataset named *Broden*, which is denoted as $\mathcal{D}_{\text{Broden}}$. The dataset contains a variety of pre-determined concepts c and images x_i with their associated pixel-level labels. Each pixel of images x_i is labeled with a set of relevant concept c , which

¹We follow previous work and use "neuron" to describe a channel in CNNs.

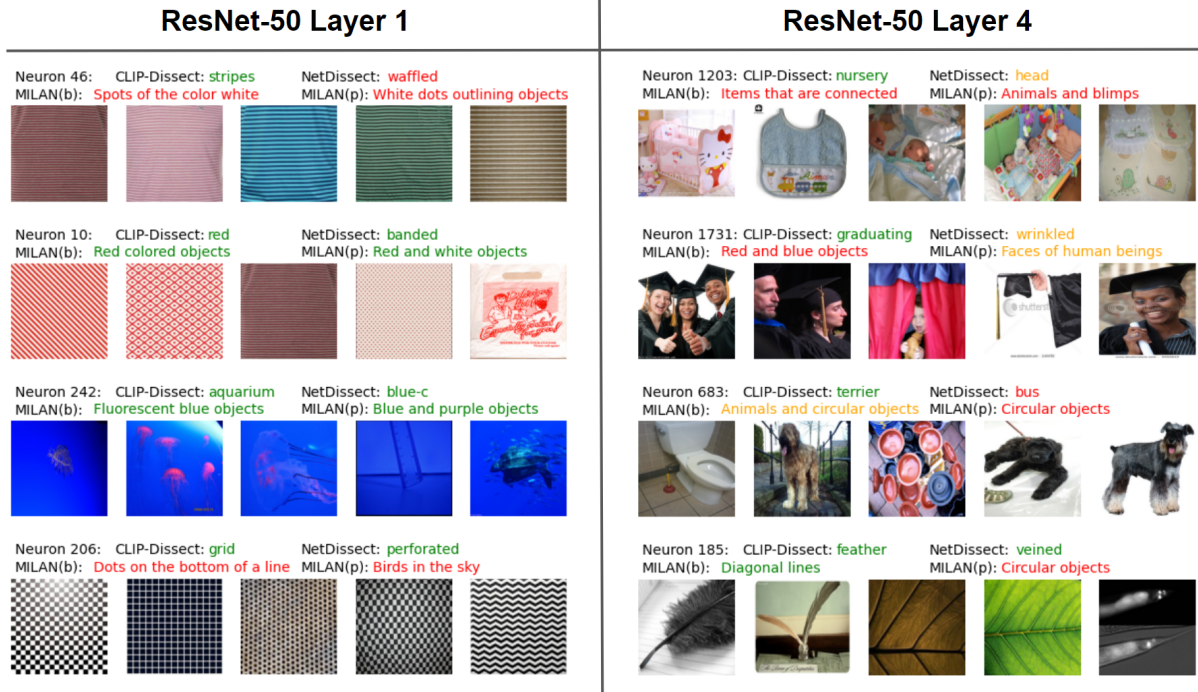


Figure 1.1. Labels generated by our method CLIP-Dissect, NetDissect [6] and MILAN [39] for random neurons of ResNet-50 trained on ImageNet. Displayed together with 5 most highly activating images for that neuron. We have subjectively colored the descriptions green if they match these 5 images, yellow if they match but are too generic and red if they do not match. In this paper we follow the *torchvision* [59] naming scheme of ResNet: Layer 4 is the second to last layer and Layer 1 is the end of first residual block. MILAN(b) is trained on both ImageNet and Places365 networks, while MILAN(p) is only trained on Places365.

provides a ground-truth binary mask $L_c(x_i)$ for a specific concept c . Based on the ground-truth concept mask $L_c(x_i)$, the authors propose to compute the intersection over union score (IoU) between $L_c(x_i)$ and the binarized mask $M_k(x_i)$ from the activations of the concerned k -th neuron unit over all the images x_i in $\mathcal{D}_{\text{Broden}}$: $\text{IoU}_{k,c} = \frac{\sum_{x_i \in \mathcal{D}_{\text{Broden}}} M_k(x_i) \cap L_c(x_i)}{\sum_{x_i \in \mathcal{D}_{\text{Broden}}} M_k(x_i) \cup L_c(x_i)}$.

If $\text{IoU}_{k,c} > \eta$, then the neuron k is identified to be detecting concept c . In [6], the authors set the threshold η to be 0.04. Note that the binary masks $M_k(x_i)$ are computed via thresholding the spatially scaled activation $S_k(x_i) > \xi$, where ξ is the top 0.5% largest activations for the neuron k over all images $x_i \in \mathcal{D}_{\text{Broden}}$ and $S_k(x_i)$ has the same resolution as the pre-defined concept masks by interpolating the original neuron activations $A_k(x_i)$.

[7] propose another version of Network Dissection, which replaces the human annotated labels with the outputs of a segmentation model. This gets rid of the need for dense annotations in D_{probe} , but still requires dense labels for training the segmentation model and the concept set is restricted to the concepts the segmentation model was trained on. For simplicity, we focus on the original Network Dissection algorithm (with human labels) in this work unless otherwise mentioned.

MILAN.

MILAN [39] is a contemporary automated neuron labeling method addressing the issue of being restricted to detect predefined concepts. They can generate unrestricted descriptions of neuron function by training a generative images-to-text model. The approach of [39] is technically very different from ours as they frame the problem as learning to caption the set of most highly activating images for a given neuron. Their method works by collecting a dataset of human annotations for the set of highly activating images of a neuron, and then training a generative model to *predict* these human captions. Thus, MILAN requires collecting this curated labeled data set, which limits its capabilities when applied to machine learning tasks outside this dataset. In contrast our method does not require any labeled data for neuron concepts and is *training-free*.

CLIP.

CLIP stands for Contrastive Language-Image Pre-training [77], an efficient method of learning deep visual representations from natural language supervision. CLIP is designed to address the limitation of static softmax classifiers with a new mechanism to handle *dynamic* output classes. The core idea of CLIP is to enable learning from practically unlimited amounts of raw text, image pairs by training an image feature extractor (encoder) E_I and a text encoder E_T simultaneously. Given a batch of N image x_i and text t_i training example pairs denoted as $\{(x_i, t_i)\}_{i \in [N]}$ with $[N]$ defined as the set $\{1, 2, \dots, N\}$, CLIP aims to increase the similarity of the (x_i, t_i) pair in the embedding space as follows. Let $I_i = E_I(x_i)$, $T_i = E_T(t_i)$, CLIP maximizes the cosine similarity of the (I_i, T_i) in the batch of N pairs while minimizing the cosine similarity of (I_i, T_j) , $j \neq i$ using a multi-class N-pair loss [89, 77]. Once the image encoder E_I and the text encoder E_T are trained, CLIP can perform zero-shot classification for any set of labels: given a test image x_1 , we can feed in the natural language names for a set of M labels $\{t_j\}_{j \in [M]}$. The predicted label of x_1 is the label t_k that has the largest cosine similarity among the embedding pairs: (I_1, T_k) .

1.3 Method

In this section, we describe CLIP-Dissect, an automatic, flexible and generalizable neuron labeling method for vision networks from popular convolutional neural networks (CNNs) to SOTA vision transformers (ViT). An overview of CLIP-Dissect algorithm is illustrated in Figure 1.2 and described in detail in Sec 1.3.1. We then introduce and discuss a few theoretically inspired choices for the similarity function in Sec 1.3.2. Finally in Sec 1.3.3 we discuss how our method can benefit from more powerful models in the future.

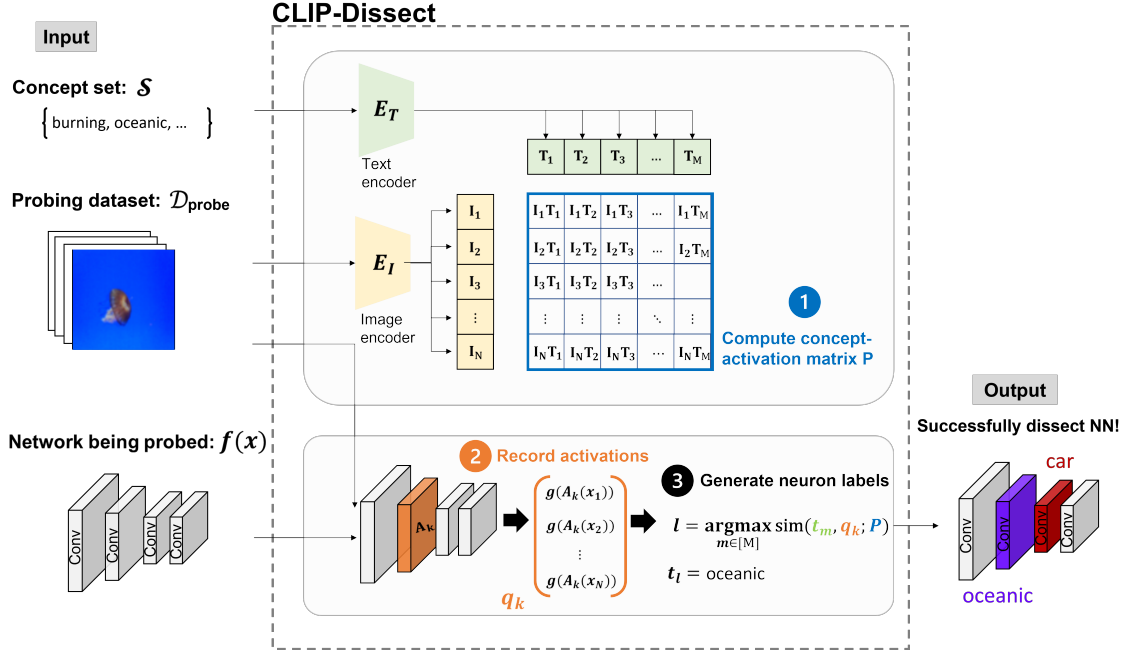


Figure 1.2. Overview of CLIP-Dissect: a 3-step algorithm to dissect neural network of interest

1.3.1 CLIP-Dissect Overview

Inputs & Outputs.

The CLIP-Dissect algorithm has 3 inputs: **(i)** DNN to be dissected/probed, denoted as $f(x)$, **(ii)** a set of probing images, denoted as $\mathcal{D}_{\text{probe}}$ where $|\mathcal{D}_{\text{probe}}| = N$, **(iii)** a set of concepts, denoted as $\mathcal{S}$, $|\mathcal{S}| = M$.

The output of CLIP-Dissect is the neuron labels, which identify the concept associated with each individual neuron. Compared with Network Dissection, our goals are the same – we both want to inspect and detect concepts associated with each neuron. The input **(i)** is also the same, as we both want to dissect the DNN $f(x)$; however, the inputs **(ii)** and **(iii)** have differences. Specifically, in CLIP-Dissect, our $\mathcal{D}_{\text{probe}}$ does not require any concept labels and thus can be any publicly available dataset such as CIFAR-100, ImageNet, a combination of datasets or even unlabeled images collected from the internet. On the other hand, Network Dissection [6] can only use a $\mathcal{D}_{\text{probe}}$ that has been densely labeled with the concepts from the concept set $\mathcal{S}$. As a result, users of Network Dissection are limited to the $\mathcal{D}_{\text{probe}} = \mathcal{D}_{\text{Brodén}}$ and the fixed concept

set $\mathcal{S}$ of Broden unless they are willing to create their own densely labeled dataset. In contrast, the concept set $\mathcal{S}$ and probing dataset $\mathcal{D}_{\text{probe}}$ in our framework are *decoupled* – we can use any text corpus to form the concept set $\mathcal{S}$ and any image dataset $\mathcal{D}_{\text{probe}}$ independent of $\mathcal{S}$ in CLIP-Dissect, which significantly increases the flexibility and efficiency to detect neuron concepts.

Algorithm.

There are 3 key steps in CLIP-Dissect:

1. *Compute the concept-activation matrix P .* Using the image encoder E_I and text encoder E_T of a CLIP model, we first compute the text embedding T_i of the concepts t_i in the concept set $\mathcal{S}$ and the image embedding I_i of the probing images x_i in the probing dataset $\mathcal{D}_{\text{probe}}$. Next, we calculate the concept-activation matrix $P \in \mathbb{R}^{N \times M}$ whose (i, j) -th element is the inner product $I_i \cdot T_j$, i.e. $P_{i,j} = I_i \cdot T_j$.
2. *Record activations of target neurons.* Given a neuron unit k , compute the activation $A_k(x_i)$ of the k -th neuron for every image $x_i \in \mathcal{D}_{\text{probe}}$. Define a summary function g , which takes the activation map $A_k(x_i)$ as input and returns a real number. Here we let g be the mean function that computes the mean of the activation map over spatial dimensions, but g can be any general scalar function. We record $g(A_k(x_i))$, for all i, k .
3. *Determine the neuron labels.* Given a neuron unit k , the concept label for k is determined by calculating the most similar concept t_m with respect to its activation vector $q_k = [g(A_k(x_1)), \dots, g(A_k(x_N))]^\top$, $q_k \in \mathbb{R}^N$. The similarity function sim is defined as $\text{sim}(t_m, q_k; P)$. In other words, the label of neuron k is t_l , where $l = \arg \max_m \text{sim}(t_m, q_k; P)$. Below we discuss different ways to define sim .

1.3.2 Similarity function

There are many ways to design the similarity function sim , and this choice has a large effect on the performance of our method. In particular, simple functions like cosine similarity

perform poorly, likely because they place too much weight on the inputs that don't activate the neuron highly. We focus on the following 4 similarity functions and compare their results in the Table 1.3:

- **Cos.** Cosine similarity between the activation vector (q_k) of the target neuron k and the concept activation matrix $P_{:,m}$ from CLIP with the corresponding concept t_m :

$$\text{sim}(t_m, q_k; P) \triangleq \frac{P_{:,m}^\top q_k}{\|P_{:,m}\| \cdot \|q_k\|} \quad (1.1)$$

- **Rank reorder.** This function calculates the similarity between q_k and $P_{:,m}$ by creating a vector q'_k , which has the values of q_k in the order of $P_{:,m}$. I.e. q'_k is generated by reordering the elements of q_k according to the ranks of the elements in $P_{:,m}$. The full similarity function is defined below, and is maximized when the q_k and $P_{:,m}$ have the same order:

$$\text{sim}(t_m, q_k; P) \triangleq -\|q'_k - q_k\|_p \quad (1.2)$$

- **WPMI (Weighted Pointwise Mutual Information).** A mathematically grounded idea to derive sim based on mutual information as used in [101], where the label of a neuron is defined as the concept that maximizes the mutual information between the set of most highly activated images on neuron k , denoted as B_k , and the concept t_m . Specifically:

$$\text{sim}(t_m, q_k; P) \triangleq \text{wpmi}(t_m, q_k) = \log p(t_m|B_k) - \lambda \log p(t_m), \quad (1.3)$$

where $p(t_m|B_k) = \prod_{x_i \in B_k} p(t_m|x_i)$ and λ is a hyperparameter.

- **SoftWPMI.** Finally, we propose a generalization of WPMI where we use the probability $p(x \in B_k)$ to denote the chance an image x belongs to the example set B_k . Standard WPMI corresponds to the case where $p(x \in B_k)$ is either 0 or 1 for all $x \in \mathcal{D}_{\text{probe}}$ while SoftWPMI relaxes the binary setting of $p(x \in B_k)$ to real values between 0 and 1.

This gives us the following function:

$$\text{sim}(t_m, q_k; P) \triangleq \text{soft_wpmi}(t_m, q_k) = \log \mathbb{E}[p(t_m | B_k)] - \lambda \log p(t_m) \quad (1.4)$$

where we compute $\log \mathbb{E}[p(t_m | B_k)] = \log(\prod_{x \in \mathcal{D}_{\text{probe}}} [1 + p(x \in B_k)(p(t_m | x) - 1)])$. As shown in our experiments (Table 1.3), we found SoftWPMI gives the best results among the four and thus we use it in all our experiments unless otherwise mentioned.

Due to page constraint, we leave the derivation and details on how to calculate WPMI and SoftWPMI using only CLIP product matrix P , as well as our hyperparameter choices to Appendix A.1.1.

1.3.3 Compatibility with future models

The current version of our algorithm relies on the CLIP [77] multimodal model. However, this doesn’t have to be the case, and developing improved CLIP-like models has received a lot of attention recently, with many recent works reporting better results with an architecture similar to CLIP [105, 107, 112, 76]. If these models are released publicly, we can directly replace CLIP with a better model without any changes to our algorithm. As a result, our method will improve over time as general ML models get more powerful, while existing works [6, 39] can’t really be improved without collecting a new dataset specifically for that purpose. Similar to ours, the segmentation version of Network Dissection [7] can also be improved by using better segmentation models, but each improved segmentation model will likely work well for only a few tasks.

1.4 Experiments

In this section, we provide both qualitative and quantitative results of CLIP-Dissect in Sec 1.4.1 and 1.4.2 respectively. We also provide an ablation study on the choice of similarity function in Sec 1.4.3 and compare computation efficiency in Sec 1.4.4. Finally, we show that

CLIP-Dissect can detect concepts that do not appear in the probing images in Sec 1.4.5. We evaluate our method through analyzing two pre-trained networks: ResNet-50 [38] trained on ImageNet [19], and ResNet-18 trained on Places-365 [115]. Our method can also be applied to modern architectures such as Vision Transformers as discussed in Appendix A.1.5. Unless otherwise mentioned we use 20,000 most common English words² as the concept set $\mathcal{S}$.

Due to the page limit, we present additional 9 experimental results in the Appendix. Specifically, Appendix A.1.2 shows additional qualitative results discussed in Section 1.4.1. Appendix A.1.3 showcases our ability to detect low-level concepts but also discusses some limitations, such as sometimes outputting higher level concepts than warranted. Appendix A.1.4 shows how our method can be applied to generate compositional concepts, and Appendix A.1.5 shows that our method can be applied to Vision Transformer architecture and provides qualitative results. In Appendix A.1.6 we experiment with another potential method to measure quality of neuron explanations and show it also favors CLIP-Dissect. Appendix A.1.7 discusses the limitations of only displaying top-5 images for qualitative evaluations and showcases a wider range for some neurons. In Appendix A.1.8 we discuss how our method can be used to decide whether a neuron is interpretable or not. Appendix A.1.9 shows the qualitative effect of different similarity functions. Finally, in Appendix A.1.10 we evaluate our description quality for 500 randomly chosen neurons, and found descriptions generated by CLIP-Dissect to be a good match for 65.5% of the neurons on average.

1.4.1 Qualitative results

Figure 1.1 shows examples of descriptions for randomly chosen hidden neurons in different layers generated by CLIP-Dissect and the two baselines: Network Dissection [6] and MILAN [39]. We do not compare against Compositional Explanations [64] as it is much more computationally expensive (at least 200 times slower) and complementary to our approach as their composition could also be applied to our explanations. We observe that not every neuron

²Source: <https://github.com/first20hours/google-10000-english/blob/master/20k.txt>

corresponds to a clear concept and our method can detect low-level concepts on early layers and provide more descriptive labels than existing methods in later layers, such as the 'graduating' and 'nursery' neurons. These results use the union of ImageNet validation set and Broden as D_{probe} . In general we observe that MILAN sometimes gives very accurate descriptions but often produces descriptions that are too generic or even semantically incorrect (highlighted as red labels), while Network Dissection is good at detecting low level concepts but fails on concepts missing from its dataset. We compared against two versions of MILAN: MILAN(b) was trained to describe neurons of networks trained on "both" ImageNet and Places365, and MILAN(p) was only trained on Places365 neurons to test its generalization ability. Additional qualitative comparisons for interpretable neurons are shown in Figures A.1 and A.2 in Appendix A.1.2.

1.4.2 Quantitative results

Besides the qualitative comparison, in this section we propose the first quantitative evaluation to compare our methods's performance with baselines. The key idea is to compare the neuron labels generated for neurons where we have access to the *ground truth* descriptions – i.e. the final layer of a network, as the ground truth concept of the output layer neuron is the name of

Table 1.1. The cosine similarity of predicted labels compared to ground truth labels on final layer neurons of ResNet-50 trained on ImageNet. The higher similarity the better. We can see that our method performs better when D_{probe} and concept set are larger and/or more similar to training data.

Method	D_{probe}	Concept set $\mathcal{S}$	CLIP cos	mpnet cos
Network Dissection (baseline)	Broden	Broden	0.6929	0.2952
MILAN(b) (baseline)	ImageNet val	-	0.7080	0.2788
CLIP-Dissect(Ours)	ImageNet val	Broden	0.7393	0.4201
	ImageNet val	3k	0.7456	0.4161
	ImageNet val	10k	0.7661	0.4696
	ImageNet val	20k	0.7900	0.5257
	ImageNet val	ImageNet	0.9902	0.9746
	CIFAR100 train	20k	0.7300	0.3664
	Broden	20k	0.7407	0.3945
	ImageNet val	20k	0.7900	0.5257
	ImageNet val + Broden	20k	0.7900	0.5233

Table 1.2. Performance when labeling final layer neurons of a ResNet18 trained on Places365. Accuracy measured on 267/365 neurons whose label is directly included in Broden labels.

Method	D_{probe}	Concept set $\mathcal{S}$	gt label annotation	Top1 Acc	CLIP cos	mpnet cos
Net-Dissect (baseline)	Broden	Broden	Yes	43.82%	0.8887	0.6697
CLIP-Dissect(ours)	Broden	Broden	No	58.05%	0.9106	0.7024

the corresponding class (class label). This allows us to objectively evaluate the quality of the generated neuron labels, which avoids the need for human evaluation and uses the real function of the target neurons while human evaluations are usually limited to describing a few most highly activating images. We propose below two metrics for measuring the quality of explanations:

- a) **Cos similarity:** We measure the cosine similarity in a sentence embedding space between the ground truth class name for the neuron (e.g. "sea lion" in Fig 1.3) and the explanation generated by the method. For embeddings, we use two different encoders: the CLIP ViT-B/16 text encoder (denoted as CLIP cos) and the all-mpnet-base-v2 sentence encoder (denoted as mpnet cos). See Figure 1.3 for an example of the similarity scores for descriptions of a single neuron.
- b) **Accuracy:** We compute accuracy for a method as the percentage of neurons that the method assigns the exact correct label i.e. the class name. Note that we only measure accuracy in situations where the method chooses from a concept set that includes the exact correct label, such as Network Dissection for models trained on Places365 (not for ImageNet models since ImageNet labels are missing from Broden). We also did not measure accuracy of MILAN as MILAN generates explanations without a concept set and thus is unlikely to match the exact wording of the class name.

In Table 1.1, we can see that the labels generated by our method are closer to ground truth in sentence embedding spaces than those of Network Dissection or MILAN regardless of our choice of D_{probe} or concept set $\mathcal{S}$. We can also see that using a larger concept set (e.g. 3k v.s. 20k) improves the performance of our method. Table 1.2 shows that our method outperforms

Final layer neuron (#150) in ResNet-50 (ImageNet), Ground truth: sea lion				
				
Methods	Generated neuron label	CLIP cos	mpnet cos	Accuracy
Network Dissection:	coast-s	0.7754	0.3473	N/A
MILAN (b):	Animals in the water	0.8037	0.3291	N/A
CLIP-Dissect(20k):	seals	0.8735	0.5151	N/A
CLIP-Dissect(ImageNet):	sea lion (exactly matches ground truth!)	1.0000	1.0000	1

Figure 1.3. Example of a final layer neuron: we compare the descriptions generated by different methods and our metrics. Accuracy only evaluated for CLIP-Dissect with ImageNet labels as concept set since it is the only method where exact correct answer is a possible choice and therefore accuracy makes sense.

Network Dissection even though this task is favorable to their method (as the Places365 dataset has large overlaps with Broden). We highlight that CLIP-Dissect can reach higher accuracy even though Network Dissection has access to and relies on the ground truth labels in Broden while ours does not.

1.4.3 Choice of similarity function

Table 1.3 compares the performance of different similarity functions used in CLIP-Dissect. We use accuracy and cos similarity in embedding space as defined in Sec 1.4.2 to measure the quality of descriptions. We observed that SoftWPMI performs the best and thus it is used in all other experiments unless otherwise mentioned. The effect of similarity function is shown qualitatively in Appendix A.1.9. Table 1.3 also showcases how CLIP-Dissect can give final layer neurons the correct label with a very impressive 95% accuracy.

1.4.4 Computational efficiency

Table 1.4 shows the runtime of different automated neuron labeling methods when tasked to label all the neurons of five layers in ResNet-50. We can see our method runs in just 4 minutes, more than 10, 60 and 200+ times faster than the baselines MILAN [39], Network Dissection [6]

Table 1.3. Comparison of the performance between similarity functions. We look at the final layer of ResNet-50 trained on ImageNet (same as Tab 1.1). We use $\mathcal{S} = 20k$ for cosine similarity and $\mathcal{S} = \text{ImageNet classes}$ for top1 accuracy. We can see SoftWPMI performs best overall.

Metric	Similarity function	D_{probe}				
		CIFAR100 train	Broden	ImageNet val	ImageNet val + Broden	Average
mpnet cos similarity	cos	0.2761	0.2153	0.2823	0.2584	0.2580
	Rank reorder	0.3250	0.3857	0.4901	0.5040	0.4262
	WPMI	0.3460	0.3878	0.5302	0.5267	0.4477
	SoftWPMI	0.3664	0.3945	0.5257	0.5233	0.4525
Top1 accuracy	cos	8.50%	5.70%	15.90%	11.40%	10.38%
	Rank reorder	36.30%	57.50%	89.80%	89.90%	68.38%
	WPMI	23.80%	47.10%	87.00%	86.90%	61.20%
	SoftWPMI	46.20%	70.50%	95.00%	95.40%	76.78%

and Compositional Explanations [64] respectively.

Table 1.4. The time it takes to describe the layers ['conv1', 'layer1', 'layer2', 'layer3', 'layer4'] of ResNet-50 via different methods using our hardware(Tesla P100 GPU). We can see CLIP-Dissect is much more computationally efficient than existing methods.

Method	CLIP-Dissect	Network Dissection	Compositional Explanations	MILAN
Runtime	3min50s	>4 hrs	>>14 hours	55min 30s

1.4.5 Detecting concepts missing from probing dataset

One surprising ability we found is that our method is able to assign the correct label to a neuron even if D_{probe} does not have any images corresponding to that concept. For example, CLIP-Dissect was able to assign the correct dog breed to 46 out of 118 neurons detecting dog breeds, and correct bird species to 22 out of 59 final layer neurons of ResNet-50 trained on ImageNet, while using CIFAR-100 training set as D_{probe} , which doesn't include any images of dogs or birds. This is impossible for any label based methods like NetDissect [6] and Compositional Explanations [64] (as IoU will be 0 for any concept not in D_{probe}), and unlikely for methods based on captioning highly activated images like MILAN [39] (as humans won't assign a caption missing from activated images). Example labels and highest activating probe

images can be seen in Figure 1.4.

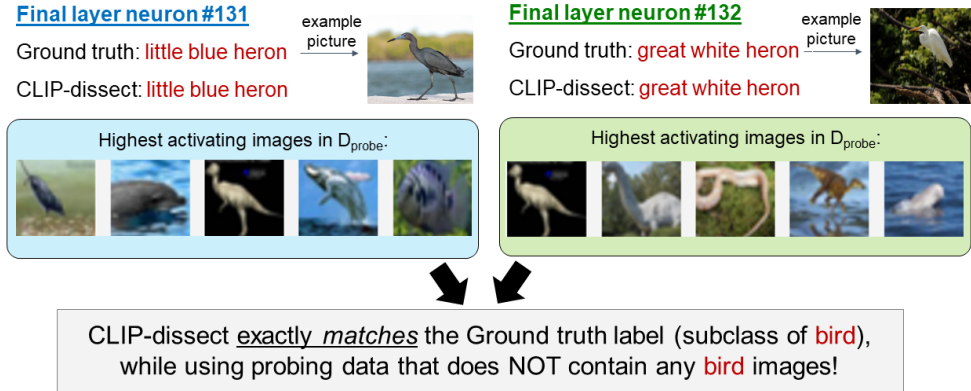


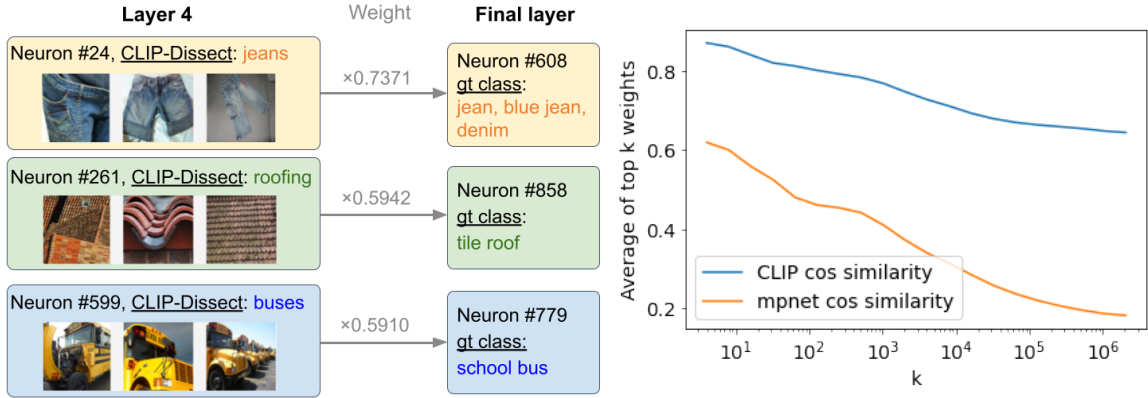
Figure 1.4. Example of CLIP-Dissect correctly labeling neurons that detect the little blue heron and the great white heron based on pictures of dolphins and dinosaurs in CIFAR. CIFAR100 does not contain any bird images but CLIP-Dissect can still get the correct concept.

1.5 Use case of CLIP-Dissect

In this section, we present a simple experiment to showcase how we can use CLIP-Dissect to gain new insights on neural networks. By inspecting the ResNet-50 network trained on ImageNet with CLIP-Dissect, we discover the following phenomenon and evidence for it: **the higher the weight between two neurons, the more similar concepts they encode**, as shown in Figure 1.5. This makes sense since a high positive weight causally makes the neurons activate more similarly, but the extent of this correlation is much larger than we expected, as each final layer neuron has 2048 incoming weights so we would not expect any single weight to have that high of an influence. A consequence of the similarity in concepts is that the second-to-last layer already encodes quite complete representations of certain final layer classes in individual neurons, instead of the representation for these classes being spread across multiple neurons. For example Fig 1.5a shows that the 3 neurons with highest outgoing weights already seem to be accurately detecting the final layer concept/class label they’re connected to.

To make these results more quantitative, in Figure 1.5b we measure the similarity of concepts encoded by the neurons connected via highest weights in the final layer of ResNet-

50. For layer4 neurons, we used CLIP-Dissect to determine their concept, while for the final layer neurons we used the ground truth i.e. class label in text form. We can clearly see that higher weights connect more similar concepts together, and the average similarity decreases exponentially as a function of k when averaging similarities of neurons connected via the top k weights. To further test this relationship, we found that the mpnet cos similarity between concepts encoded by two neurons and the weight connecting them are correlated with $r = 0.120$ and p-value $< 10^{-300}$ (probability of no correlation is practically 0) when calculated over all 2 million weights in the final layer. If we only look at the highest 50000 weights, the correlation is even higher with $r = 0.258$, p-value $< 10^{-300}$.



(a) Visualization of 3 highest weights of final layer. (b) Average cosine similarity between concepts.

Figure 1.5. a) 3 highest weights of the final layer of ResNet-50 trained on ImageNet, we can see neurons connected by the highest weights are detecting very much the same concept. b) Cosine similarities between the concepts of neurons connected by highest weights. The higher the weight between neurons, the more similar a concept they represent.

1.6 Limitations and conclusions

Limitations: The main limitation of our method compared to previous work is that it's not taking advantage of the spatial information of neuron activations. This causes some difficulties in detecting lower level concepts, but we are still able to detect many low level/localized patterns as discussed in Section A.1.3. Secondly, our method currently works well only on concepts and images that CLIP works well on, and while this already covers a larger set of tasks than

what existing neuron labeling methods perform well on, CLIP-Dissect may not work out of the box on networks trained on tasks that require highly specific knowledge such as classifying astronomical images. However, our method is compatible with future large vision-language models as long as they share a similar structure to CLIP, whether they are general models or trained for a specific domain. Finally, not all neurons can be described well by simple terms such as single word explanations. While we can augment the space of descriptions using a different concept set $\mathcal{S}$, or creating compositional explanations as discussed in Appendix A.1.4, some neurons may have a very complicated function or perform different functions at different activation ranges. For the most part, current methods including ours will be unable to capture this full picture of complicated neuron functions.

Conclusions: In this work, we have developed CLIP-Dissect, a novel, flexible and computationally efficient framework for automatically identifying concepts of hidden layer neurons. We also proposed new methods to quantitatively compare neuron labeling methods, which is based on labeling final layer neurons. We have shown CLIP-Dissect can match or outperform previous automated labeling methods both qualitatively and quantitatively and can even detect concepts missing from the probing dataset. Finally we used CLIP-Dissect to discover that neurons connected by a high weight often represent very similar concepts.

1.7 Acknowledgements

The authors would like to thank anonymous reviewers for valuable feedback to improve the manuscript. The authors also thank MIT-IBM Watson AI lab for computing support in this work. T. Oikarinen and T.-W. Weng are supported by National Science Foundation under Grant No. 2107189.

Chapter 1, in full, is based on the paper from Tuomas Oikarinen, and Tsui-Wei Weng "CLIP-Dissect: Automatic Description of Neuron Representations in Deep Vision Networks" published at the International Conference on Learning Representations (ICLR), 2023. The

dissertation author is the primary investigator and author of this paper.

Chapter 2

Linear Explanations for Individual Neurons

2.1 Introduction

Current machine learning models are extremely capable at many tasks, yet they are notoriously hard to understand, and most often seen as black boxes. Recently, the field of *mechanistic interpretability* [72, 22] has emerged to address this issue by providing mechanistic understanding of the inner workings of neural networks, with the ultimate goal of reverse engineering the algorithms that neural networks use to solve problems.

Individual neurons (or channels of a CNN) are the most simple unit of a neural network, and understanding them is a fundamental building block of mechanistic interpretability. Many methods have been developed to understand individual neurons, based on both manual inspection [23, 114, 72] and automated description [6, 39, 68]. However, most of these methods only focus on understanding and explaining the very highest activations of a neuron. In Section 2.2 we show that this is not enough, and a neuron’s impact on network outputs is distributed quite evenly across all inputs. Therefore, to fully understand a neuron, we need to understand everything it is doing, not just its highest activations.

In Section 2.3 we propose a solution to these issues, which we call **Linear Explanations (LE)**. In our method, each neuron is described as linear combination of concepts, such as “ $w_1 \times \text{dog} + w_2 \times \text{pet}$ ”. This explanation allows us to directly simulate neuron activations as

$s(x_i) = w_1 \mathbb{P}(\text{dog}|x_i) + w_2 \mathbb{P}(\text{pet}|x_i)$, where $\mathbb{P}(\text{concept}|x_i)$ can be estimated by either a human or a model. Our automated method allows for learning accurate linear explanations efficiently, as demonstrated in Section 2.5, where we evaluate and compare our method and existing automated explanation methods. We believe linear explanations are a natural way to represent neurons, and they can elegantly model the scalar nature of a neuron’s activation. In addition, linear explanations can be found efficiently and do not binarize neuron activations (which loses information), unlike alternative methods to capture complex neuron behavior such as Compositional Explanations [65].

In Section 2.4 we propose a new, more rigorous way to automatically evaluate the quality of neuron descriptions via *simulation*, i.e. predicting the neuron’s activation on a new input given the explanation. The simulation evaluation was recently proposed for explaining neurons of large language models by [9], and has since been a popular evaluation method in language settings [18, 14]. We are the first to present an effective and natural way to run simulation on vision models by utilizing SigLIP [111] models as the simulator. We find that existing neuron explanations [6, 39, 68] perform poorly under simulation, while our method provides more than $3\times$ higher ablation scores when simulated. Finally, in Figure 2.3 we show how our method can uncover multiple roles played by the same neuron that would be missed when only looking at highest activations. Our code and results are available at <https://github.com/Trustworthy-ML-Lab/Linear-Explanations>.

2.2 Motivation: How Important are Different Parts of a Neuron’s Activation Pattern?

In this section, we aim to answer the following question:

Is most of a neuron’s impact on the network caused by the very highest activating inputs, or are all inputs important?

We do this by ablating the individual neurons of a network, i.e. replacing their activation

by 0, and measuring the change in network outputs. We refer to this as the neuron’s causal effect.

2.2.1 Definitions

To describe our results, we first need to define a few metrics: The idea is to measure the causal effect of these neurons in terms of metrics most relevant to the end use case, which in a classification setting are accuracy and cross-entropy loss.

Let $f(\cdot)$ be a neural network of interest. $f(x)$ is the network’s output on an input x , i.e. class probabilities. Let $D = \{(x_i, y_i)\}$ be a dataset with images x_i and corresponding ground-truth class labels y_i . We define $I_k(x_i, y_i)$ as the impact of neuron k on the input x_i :

$$I_k(x_i, y_i) := \frac{\Delta \text{Acc}_k(x_i, y_i) - \Delta L_k(x_i, y_i)}{2} \quad (2.1)$$

where $\Delta \text{Acc}_k(x_i, y_i)$ is the change in accuracy and $\Delta L_k(x_i, y_i)$ is the change in loss as formally defined below:

$$\Delta \text{Acc}_k(x_i, y_i) = \frac{[h(f(x_i)) = y_i] - [h(f_{\sim k}(x_i)) = y_i]}{\sum_{(x_j, y_j) \in D} [h(f(x_j)) = y_j]} \quad (2.2)$$

$$\Delta L_k(x_i, y_i) = \frac{L(f(x_i), y_i) - L(f_{\sim k}(x_i), y_i)}{\sum_{(x_j, y_j) \in D} L(f(x_j), y_j)} \quad (2.3)$$

Here $[\cdot]$ is the indicator function, taking value 1 if the expression is True and 0 otherwise, $h(\cdot) := \text{argmax}(\cdot)$, $f_{\sim k}(\cdot)$ is the output of the model without neuron k , i.e. the neuron’s activation is replaced by 0 and L is the loss function, such as cross-entropy loss. We use temperature calibration [35] before calculating the losses.

The impact $I_k(x_i, y_i)$ is then the average of the neuron’s effect on accuracy and loss of the model on an input, as a fraction of the model’s total loss and accuracy, with signs chosen such that a positive impact means including the neuron k improves the network’s predictions.

Next, we measure how much of a neuron’s total impact is caused by the inputs that activate it the highest. To achieve this, we define *Top Impact*, denoted as $TI_k(\beta)$, with the input

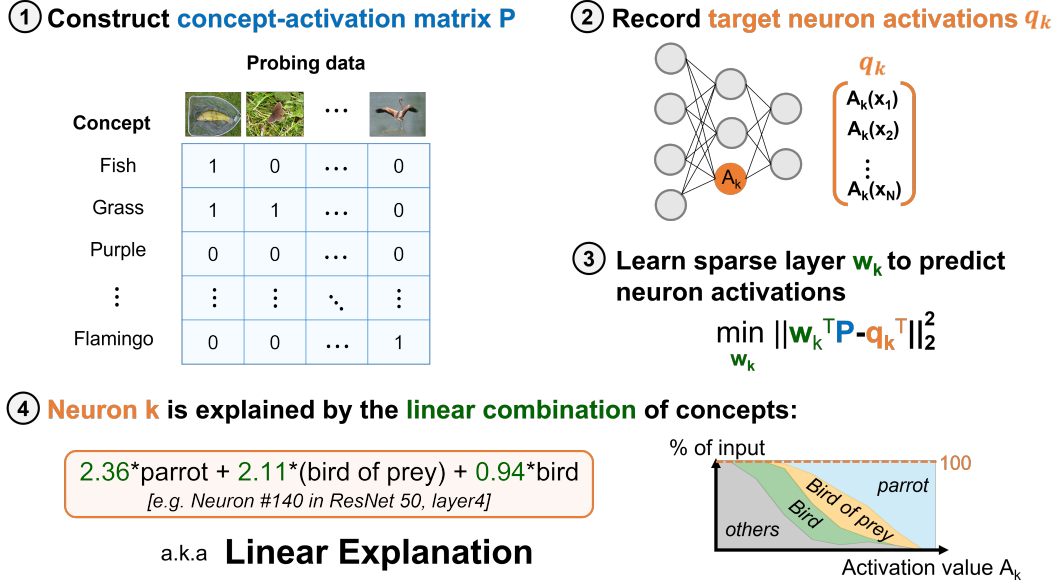


Figure 2.1. Overview of our proposed method: Linear Explanations.

argument $\beta \in [0, 1]$ representing the fraction of highest activating inputs included:

$$TI_k(\beta) = \frac{\sum_{i=1}^{\beta|D|} |I_k(x_{(i)}, y_{(i)})|}{\sum_{(x_j, y_j) \in D} |I_k(x_j, y_j)|} \quad (2.4)$$

where $x_{(i)}, y_{(i)}$ are ordered in descending order of neuron k 's activation. I.e. $g(A_k(x_{(i)})) \geq g(A_k(x_{(i+1)})) \forall i$, where $A_k(x_i)$ is the activation of a neuron or a channel in CNNs on input x_i , and g is a summary function such as mean or max that takes the 2D activation map of a CNN channel into a single scalar (or identity for scalar neurons) as defined in [68]. A neuron where all inputs are equally important should have $TI_k(\beta) = \beta$.

2.2.2 Results

To measure how important highly activating inputs are to network predictions in practice, we ablated out all the neurons (channels) of ResNet-50 [38] one at a time and measured the change in performance on ImageNet validation data. Results in terms of *Top Impact* are shown in Table 2.1. We used mean as summary function g . We can see that if we wish to understand most of a neuron's impact (i.e. high TI), we need to look at a large fraction of the neuron's inputs

(high β), not just the most highly activating inputs.

This is in contrast to many popular methods of single neuron explanation, which often focus exclusively on the most highly activating inputs, i.e. very small β . For example MILAN [39] only looks at 15 most highly activating inputs, which is equivalent to $\beta = 0.0003$. In Table 2.1, we can see that these inputs only explain 0.258% of the neuron’s impact on average. Similarly, CLIP-Dissect [68] with SoftWPMI similarity function only looks at 100 ($\beta = 0.002$) most highly activating inputs, which make up 1.522% of the neuron’s impact. While Network Dissection [6] looks at all inputs, it only aims to explain top 0.5% of location specific neuron activations which causes similar issues. Based on the findings in Tab 2.1, we believe most inputs of a neuron are important, and only focusing on highest activating inputs (small β) is not sufficient to faithfully explain individual neurons, or to evaluate how good such explanations are. To resolve this issue, we propose a new explanation method (Section 2.3) and a new evaluation metric (Section 2.4) that focus on explaining the *entire* range of a neuron’s activations.

2.3 Method

In this section we present our method **Linear Explanations** or **LE**. Our method consists of two main parts: (i) Constructing a concept activation matrix P , and (ii) Learning a linear combination to explain the neuron. An overview of our method is shown in Figure 2.1.

Table 2.1. Average *Top Impact* for neurons in different layers of ResNet-50 (ImageNet). We can see that while highly activating inputs (small β) are more impactful than the average input, especially on later layers, they still explain only a small portion of the neuron’s total effect. E.g. $\beta = 0.002$ only accounts for around 1.52% of Total Impact.

β	Conv 1	Layer 1	Layer 2	Layer 3	Layer 4	All
0.0003	0.054%	0.084%	0.110%	0.268%	0.319%	0.258%
0.002	0.319%	0.454%	0.574%	1.369%	2.007%	1.522%
0.02	2.743%	3.526%	4.160%	8.654%	13.59%	10.22%
0.1	12.42%	14.80%	16.53%	28.28%	39.38%	31.42%
0.5	55.23%	59.49%	62.08%	76.85%	84.55%	77.46%

2.3.1 Constructing a concept activation matrix

An essential part of our explanation is creating the concept activation matrix P . Each entry P_{ij} represents how much of concept c_i is present in input x_j in a way that is aligned with human perception. That is $P_{ij} \approx \mathbb{P}(c_i|x_j)$, for each input x_j of the probing dataset $\mathcal{D}_{\text{probe}}(|\mathcal{D}_{\text{probe}}| = N)$ and each concept c_i in the concept set $\mathcal{S}(|\mathcal{S}| = M)$.

In this paper we experiment with two different methods of determining P described below: **Label** and **SigLIP**.

Label: Use labeled data. If we have access to labeled data for all concepts in the concept set for all inputs in the probing dataset, we can directly use these labels as our concept activation matrix P . This is the ideal case and works in situations where such data is available, such as the Broden [6] dataset or the class labels on a validation dataset. We denote this approach as **LE(Label)**. However, we often do not have access to labels for all concepts and/or images we want to utilize, and collecting additional labeled data can be expensive. This leads us to our second method **SigLIP**.

SigLIP: Pseudo-labels from Multimodal Models. In cases where sufficient labeled data is not available, it is often beneficial to create artificial labels using pretrained foundation models such as CLIP [77]. In our experiments we used SigLIP [111](ViT-L-16-384) as our *explainer model*. SigLIP(ViT-L-16-384) is a more recent model similar to CLIP, and was chosen because of its improved performance, as well as use of sigmoid function during pretraining. Training with sigmoid activation is important for creating a reliable concept activation matrix P , as it is essentially a multi-label classification task, i.e. most images contain multiple concepts. Because of this, softmax based models will likely perform poorly on constructing a concept activation matrix P .

To ensure the pretrained SigLIP model is aligned with human concept predictions, we add additional calibration parameters a and b and optimize these using publicly available data. Let E_I be the image encoder and E_T be the text encoder of the SigLIP model. We generate

concept activation matrix P as follows: $P_{ij} = \sigma(a \cdot E_T(c_i)E_I(x_j) + a \cdot b)$. Since our goal is to have $P_{ij} \approx \mathbb{P}(c_i|x_j)$, i.e. to be similar to human predictions, we learn a and b by minimizing binary cross-entropy loss L_{BCE} on concepts we have labels for. In particular we use ImageNet validation data, with added superclass labels according to WordNet hierarchy [61], which turns this into a multi-label classification task. a and b are then determined as:

$$\min_{a,b} \sum_{x_i \in D} \sum_{c_j \in C} L_{BCE}(\sigma(a \cdot E_T(c_j)E_I(x_i) + a \cdot b), y_{i,j}) \quad (2.5)$$

where C is the set of [super]class names, c_j is the name of j -th [super]class, $y_{i,j}$ is a binary label indicating whether input x_i belongs to [super]class c_j and L_{BCE} is binary cross-entropy loss. The hyperparameters a and b are explainer model specific, and independent of $\mathcal{D}_{\text{probe}}$ or $\mathcal{S}$.

2.3.2 Learning Linear Explanations

Once we have our concept activation matrix $P \in \mathbb{R}^{M \times N}$, our next task is to learn a sparse linear model that can predict the activation of our target neuron based on the presence of a few concepts in our input. We do this in two steps described below. Let $q_k \in \mathbb{R}^N$ be the activation

vector of neuron k , defined as $q_k = \begin{bmatrix} g(A_k(x_1)) \\ g(A_k(x_2)) \\ \vdots \\ g(A_k(x_N)) \end{bmatrix}$, where $A_k(x_i)$ is the activation of neuron k on

input x_i , and $g(\cdot)$ is a summary function (mean) which is used in case the neuron's activation is not a scalar, for example when neuron k is a channel of a CNN.

Throughout the paper we use a 70-10-20 split to divide our $\mathcal{D}_{\text{probe}}$ into train, validation and test set splits to avoid overfitting our explanations. We denote the train subset of q_k and P as q_k^{train} and P^{train} respectively.

Learn a relatively sparse w_k

First, we use the GLM-Saga package [103] to learn a sparse linear weight $w_k \in \mathbb{R}^M$ to minimize the following objective:

$$\mathcal{L}_{\text{MSE}} = \|w_k^\top P^{\text{train}} - (q_k^{\text{train}})^\top\|_2^2 + \lambda R_\eta(w_k) \quad (2.6)$$

where $R_\eta(w_k) = (1 - \eta)\frac{1}{2}\|w_k\|_2^2 + \eta\|w_k\|_1$ and η is a hyperparameter, set to 0.99 in our experiments. It is worth noting that we optimize the predictions to be accurate on the entire $\mathcal{D}_{\text{probe}}^{\text{train}}$ (i.e. $\beta = 1$), so our explanations describe the entire activation range, not just the most highly activating inputs as discussed in Sec 2.2. Our goal is to learn w_k that can accurately predict neuron activations based on just the concept information embedded in the concept activation matrix P , while also being sparse for interpretability. The sparsity constraint is reflected in the term $\|w_k\|_1$ of the regularization $R_\eta(w_k)$. This is a surrogate term for the exact sparsity goal of minimizing the ℓ_0 norm (i.e. $\|w_k\|_0$), as the ℓ_1 norm is convex and much easier to optimize than the non-convex ℓ_0 norm. However we found it hard/unstable to find extremely sparse (around 5 concepts per neuron) w_k using only this method.

Next we discuss how to overcome this using greedy search guided by the found values w_k . For discussion on results without using greedy search, see Appendix B.2.4.

Greedy search

Our basic idea for greedy search is to use the weights w_k found in previous step as a heuristic to find a very sparse explanation for the neuron of interest. This greedy-search process is described in detail in Algorithm 1 and Appendix B.1.3. For each neuron, we test $r = 10$ available concepts with the highest weight, and choose the one that can train the best model together with the already selected concepts. We continue this process until we reach the maximum number of concepts $v = 10$, or until adding another concept does not improve performance enough. This is modeled by the tolerance parameter ε . This lets our method dynamically decide the number of

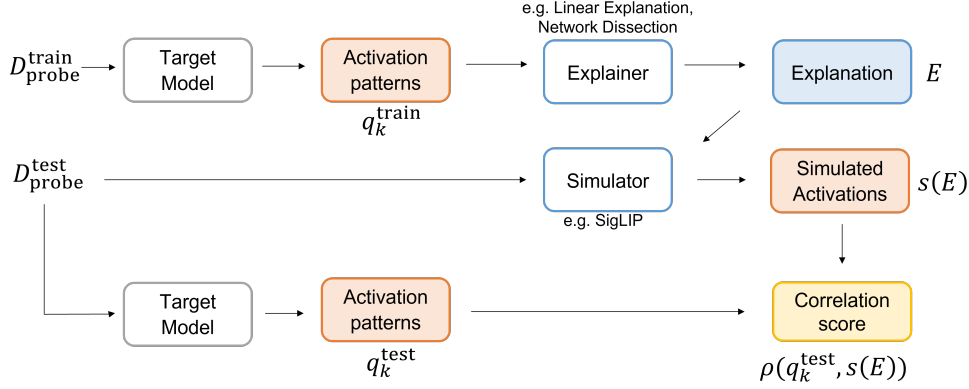


Figure 2.2. An overview of the simulation pipeline with correlation scoring.

concepts for each neuron, adding more complexity to the explanation only when it is needed. The tolerance parameter can be adjusted to make a tradeoff between simpler and shorter v.s. more complete explanations. This process requires training many models per each neuron, but they are extremely small linear models (10 or less parameters) which can be trained to optimal in a fraction of a second. Each of these models is trained to minimize MSE loss of predicting neuron activations as a linear function of only the selected concepts (with no regularizer), i.e. the first term in equation 2.6. More concretely, given set of concept indices discovered from Greedy search $\mathcal{J} = \{i_1, \dots, i_u\}$, $u \leq v$, our final explanation E is then:

$$E = \{(w_{k,1}^*, c_{i_1}), \dots, (w_{k,u}^*, c_{i_u})\} \quad (2.7)$$

where $w_k^* = \operatorname{argmin}_{w \in \mathbb{R}^u} \|w^\top P_{\mathcal{J}}^{\text{train}} - (q_k^{\text{train}})^\top\|_2^2$.

2.4 Improving Evaluation of Explanations via Simulation

An important part of creating explanations for individual neurons is being able to evaluate how faithfully these descriptions actually correspond to target neuron behavior. This has traditionally been done with methods such as evaluating whether the description matches the most highly activating images [6, 39, 68, 46], but this only evaluates a very small portion of the neuron’s activations, which is not sufficient for understanding the neuron as we have shown in

Section 2.2. In addition, [117] showed that while the very highest activations of a neuron may often be understandable, understanding a larger part of the activation range quickly becomes difficult for humans.

Inspired by recent work in language models [9], we instead propose to evaluate our explanations using *simulation*. See Figure 2.2 for an overview of our simulation pipeline. The basic idea of *simulation* is as follows:

1. Generate a human understandable explanation E for the neurons using an *Explainer*¹.
2. Use a *Simulator* to predict neuron activations $s(x, E)$ on new inputs x , based on only the explanation E and the input.
3. Score how well the simulated activations fit the actual neuron activations, using either correlation or ablation scoring.

The intuition behind simulation is that a good explanation should allow the simulator model (or a human) to accurately predict the neuron activations. [9] used GPT-4 as both the explainer and the simulator, showing the 5 most highly activating text excerpts from a subset of its training data to generate an explanation, and then simulate its activations on 5-10 new excerpts to evaluate how good these explanations are.

In this section, we propose how to naturally extend the idea of simulation from language to the vision domain, and discuss some conceptual and practical improvements we can do in this modality. We argue a good simulator should have the following properties:

1. The simulator needs to be able to take any textual concept, and predict how highly it activates on any given image (or subpart of the image).
2. The simulator and the explainer should be **different** models. Using the same model may overestimate the quality of the explanations. For example, the explainer could use an

¹Explainer could be a neural network, a human or an algorithm using human-annotated labels

uninterpretable code, or a misunderstood concept to explain a neuron and still receive a high score if the simulator shares this misunderstanding.

3. Third, a simulator should be *human aligned* i.e. predict activations similar to how a human would.

Following the above, we propose using CLIP [77] or similar models to perform this image level simulation. In particular, we choose the SigLIP-SO400M-14-384 [111] as the simulator, because it is the most powerful sigmoid trained model available. Note this is a different SigLIP model than the one we use to generate explanations of **LE(SigLIP)**.

To ensure that our simulator is human-aligned, we base our simulation on predicting $\mathbb{P}(c_i|x_j)$, i.e. how likely is the concept represented by c_i to be present in image x_j . This is important as it is aligned with how humans usually think about concepts, instead of directly predicting real values as done by [9]. We estimate $\mathbb{P}(c_i|x_j)$ similar to how we constructed matrix P in Section 2.3, i.e.

$$\mathbb{P}_{sim}(c_i|x_j) = \sigma(a_{sim}E_T^{sim}(c_i)E_I^{sim}(x_j) + a_{sim}b_{sim}) \quad (2.8)$$

Like before, we optimize the hyperparameters a_{sim} and b_{sim} on ImageNet validation data with superclasses as defined in equation 2.5. Different from the explainer (Section 2.3), the simulator does not use a fixed concept set, but instead evaluates all the concepts present in explanation E .

For explainers that produce a single text explanation c_e [6, 39, 68], their explanation can be written as a linear explanation of length 1, i.e. $E = \{(1, c_e)\}$. Once we have an explanation E , the initial simulated activation $s(x_j, E)$ is calculated in the following way:

$$s(x_j, E) = \sum_{(w_i, c_i) \in E} w_i \mathbb{P}_{sim}(c_i|x_j) \quad (2.9)$$

With $s(x_j, E)$, we can evaluate the quality of explanation E using two scoring methods:

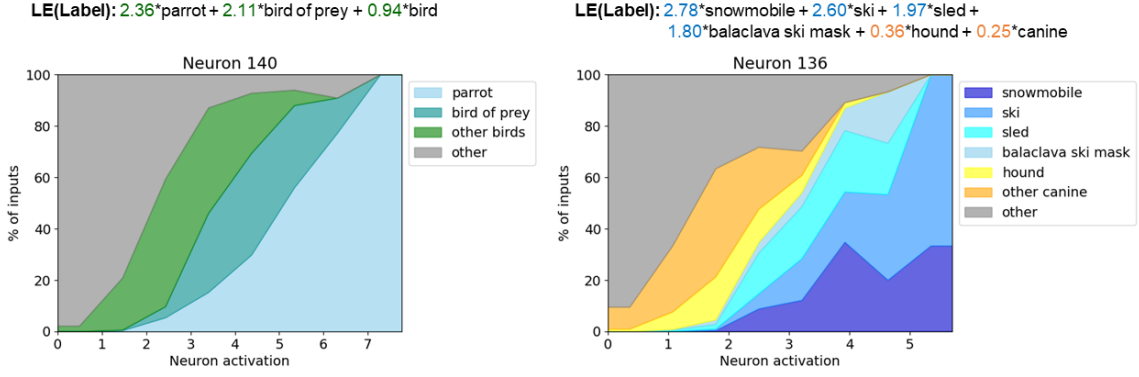


Figure 2.3. An area chart of the activations of two neurons in layer4 of ResNet-50. We can see Neuron 140 is mostly monosemantic, but represents different types of birds at different activation ranges. In contrast, neuron 136 has two distinct roles, snow and skiing related concepts at high activations and dog-like animals at lower activations.

A. Correlation Scoring $\rho(k, E)$: The explanations E are scored based on the correlation coefficient between the simulated activations and the neuron’s real activation patterns.

$$\rho(k, E) = \sum_{x \in \mathcal{D}_{\text{probe}}^{\text{test}}} \frac{\hat{s}(x, E) \cdot \hat{g}(A_k(x))}{|\mathcal{D}_{\text{probe}}^{\text{test}}|} \quad (2.10)$$

where $\hat{s}$ and $\hat{g}(A_k)$ are s and $g(A_k)$ normalized to have mean 0 and standard deviation of 1 on the test distribution.

B. Ablation scoring $\alpha(k, E)$: In ablation scoring, we directly replace the actual neuron activation with our simulated neuron activations, and measure how much this changes the final model outputs. Unlike correlation scoring, for ablation scoring we need to make sure simulated activations have the right scale. This leads us to ablation simulated value $s_{abl}(x, E, c, d)$, where c and d are scaling parameters used to match the magnitude of predicted activations with actual neuron activations:

$$s_{abl}(x, E, c, d) = c \cdot s(x, E) + d \quad (2.11)$$

We measure ablation performance with an objective adapted for classification setting, defined as

$$\alpha_{init}(k, \mathcal{D}, E, c, d) =$$

$$1 - \frac{\sum_{(x,y) \in \mathcal{D}} |L(f_{k \leftarrow s_{abl}(x,E,c,d)}(x), y) - L(f(x), y)|}{\sum_{(x,y) \in \mathcal{D}} |L(f_{k \leftarrow \mu}(x), y) - L(f(x), y)|} \quad (2.12)$$

where $f(x)$ is the output of the target model, L is the loss function (e.g. cross-entropy-loss), $f_{k \leftarrow s_{abl}(x,E,c,d)}(x)$ indicates replacing the activations of neuron k with the simulated values $s_{abl}(x, E, c, d)$, while $f_{k \leftarrow \mu}(x)$ means replacing the neuron k 's activation with its mean value. This is the same as the ablation objective of [9], except we have replaced Jensen-Shannon divergence with cross-entropy loss as it is a more relevant metric in the classification task. Note this formulation requires using $\mathcal{D}$ where we have access to ground truth labels, which is the case when using validation data.

To get the parameters c, d , we optimize with gradient descent on the validation split:

$$c^*, d^* = \arg \max_{c,d} \alpha_{init}(k, \mathcal{D}_{probe}^{val}, E, c, d) \quad (2.13)$$

Our final simulation score is then evaluated on the test split, and defined as:

$$\alpha(k, E) = \alpha_{init}(k, \mathcal{D}_{probe}^{test}, E, c^*, d^*) \quad (2.14)$$

A perfect simulation will reach $\alpha(k, E) = 1$, while random guess should receive a score of 0.

Here we used optimization to find the parameters c and d . [9] instead calculated c and d based on correlation between the predicted and true activation. We evaluate the difference between these choices in Appendix B.2.5, and find our optimization method gives noticeably higher ablation scores, but with a higher computational cost.

Finally we note that our SigLIP based simulation pipeline is much more computationally efficient than the GPT-4 pipeline of [9] because we do not need to recalculate image embeddings when simulating a new explanation. This allows us to run simulation on the entire test split of probing data (10,000 images for ImageNet), which is many orders of magnitude larger than the

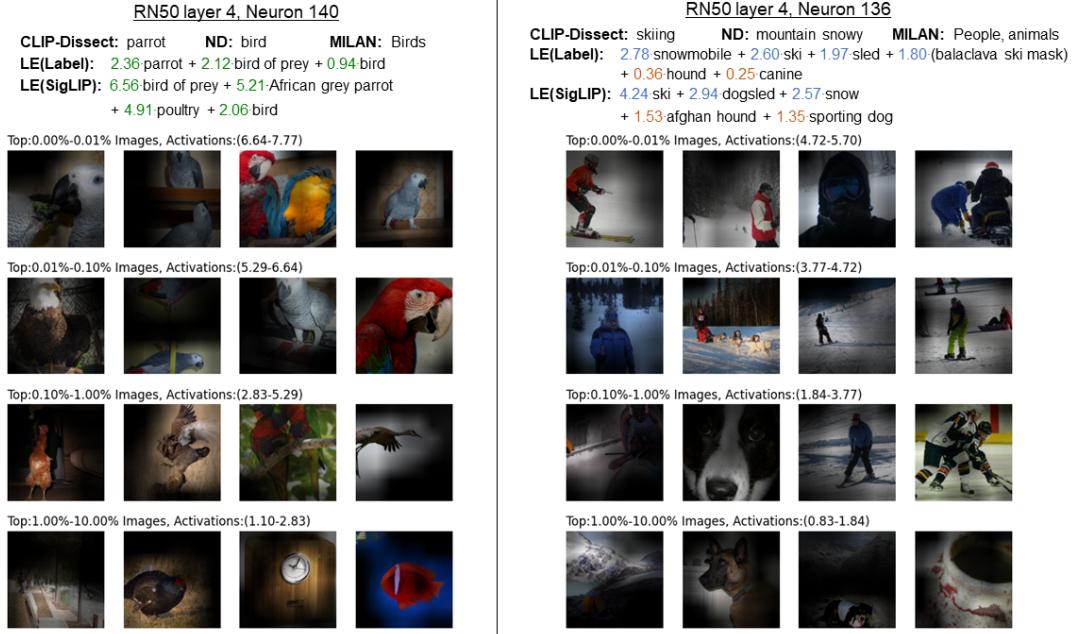


Figure 2.4. Descriptions and highly activating images from different ranges of example neurons. We can see Linear Explanation provides a more complete description than baselines in both cases.

number of samples [9] used for simulations.

2.5 Experiment Results

Setup

We mostly focus our analysis on second to last layer features similar to [46] and [16] because they are the highest level features learned by the model, and are the features used when transfer learning from that model. Additionally, in the ResNet [38] family of models, they are CNN channels followed by a global avg pooling layer, meaning they can be meaningfully understood (and simulated) either as having 2d or scalar activations without losing any information.

We evaluate two variants of our method: **Linear Explanation (Label)**, which uses the labels in $\mathcal{D}_{\text{probe}}$ to construct P , as well as superclass labels in the case of ImageNet and CIFAR-100. The second variant is **Linear Explanation (SigLIP)**, which uses SigLIP-ViT-L-16-384

model (different from our simulator model) to construct the concept activation matrix P .

$\mathcal{D}_{\text{probe}}$: As probing data we use the validation dataset of the dataset the model was trained on. We randomly split this probing data into train(70%), validation(10%) and test(20%) splits. Since we simulate on the entire test split, our setting corresponds to the most challenging *random-only* setting from [9].

Concept set $\mathcal{S}$: For **LE (Label)**, the concept set is the label names in the original dataset (+ superclass names). For **LE(SigLIP)**, we use union of the label names, the labels in Broden [6], and a list of 6800 English nouns. For SigLIP we filtered these concepts to only use concepts whose average top-5 activation in $\mathcal{D}_{\text{probe}}$ was ≥ 0.5 to avoid using concepts not present in the data. See Appendix B.2.2 for an ablation study using different concept sets.

2.5.1 Qualitative results

In Figure 2.3, we display example **LE(Label)** explanations, as well as an area chart visualization for a few example neurons. This choice of visualization was inspired by [31], as it can visualize the neuron behavior across the entire activation range, and with our label based method we can construct them automatically. To construct this, we divided neuron activations into 8 evenly spaced buckets between 0 and the max activation of the neuron, and then plot fraction of inputs within that activation range that belong to each class/superclass. Note the later buckets have much fewer inputs in them. We can see that these neurons are very well explained by the Linear Explanation, and LE can help reveal polysemanticity across different activation ranges, such as Neuron 136 which activates on dog related concepts on lower activations (orange region), while its top activations are snow related concepts (blue region). Existing methods miss the dog related role and only explain the top activations, as seen in Figure 2.4. We display explanations from ours and other methods in a more traditional way by visualizing inputs from different activation ranges in Figure 2.4, and for many more neurons in Appendix B.3.3.

Table 2.2. Average correlation scores between simulated and actual neuron activations, across all neurons in the second to last layer of the respective models. For ViT models we report the MLP neurons in the last transformer block.

Target model	Net Dissect	MILAN	CLIP-Dissect	LE (Label)	LE (SigLIP)
ResNet-50 (ImageNet)	0.1242 ± 0.002	0.0920 ± 0.002	0.1871 ± 0.002	0.2924 ± 0.002	0.3772 ± 0.002
ResNet-18 (Places365)	0.2038 ± 0.005	0.1557 ± 0.005	0.2208 ± 0.005	0.3388 ± 0.004	0.4372 ± 0.003
VGG-16 (CIFAR-100)	n/a	n/a	0.2298 ± 0.004	0.4330 ± 0.004	0.4970 ± 0.004
ViT-B/16 (ImageNet)	n/a	n/a	0.1722 ± 0.004	0.3243 ± 0.005	0.3489 ± 0.005
ViT-L/32 (ImageNet)	n/a	n/a	0.0549 ± 0.002	0.1879 ± 0.004	0.2182 ± 0.004

Table 2.3. Average Ablation scores between simulated and actual neuron activations, average over all neurons in layer4.

Target model	Network Dissection	MILAN	CLIP-Dissect	LE(Label)	LE(SigLIP)
ResNet-50 (IN)	0.0165 ± 0.0003	0.0137 ± 0.0003	0.0215 ± 0.0004	0.0433 ± 0.0007	0.0727 ± 0.0008

2.5.2 Simulation results: Correlation Scoring

Table 2.2 shows the average correlation scores between simulated and actual neuron activations, across many different models and network architectures, including both CNNs and ViTs. We do not include Network Dissection and MILAN results for the last three models, as those methods are designed for 2d activations, while the final layers of these models have effectively scalar activations. We can see Linear Explanations, especially SigLIP, significantly outperform existing methods, reaching around 0.4 correlation on average, twice as high as existing methods. There is relatively large variance in how interpretable individual units are between different architectures and datasets, but performance between methods is quite consistent. The average length of our LE explanations are shown in Table B.1(App. B.1.4). Correlation scores for other layers of a network are discussed in Appendix B.3.1. In Table B.2(App. B.2.1) we perform an ablation study by using a different simulator model, showing that the trends we observe here hold across different simulators.

2.5.3 Simulation results: Ablation scoring

In Table 2.3, we score the explanations using ablation scoring, reporting the average score across all neurons of layer 4 in ResNet-50. We find that existing methods perform very poorly under ablation scoring, with all methods averaging scores of at most 0.02, where simply predicting mean activation on all inputs would give a score of 0, and a perfect prediction can reach 1. Our methods improve significantly over existing methods, with Linear Explanation (SigLIP) reaching 0.0727 average ablation score, more than $3 \times$ better than previous methods, but still scoring quite low overall. This highlights the need for further refinement in neuron explanation methods. This is consistent with results of [9] in language models, where they found most neuron explanations scored very close to 0 under random-only ablation scoring. We also found that there is a clear quadratic relationship between the correlation and ablation score of an explanation across all explanation methods, as shown in Figure 2.5. This shows correlation score is predictive of ablation score.

Discussion.

Overall we found that **LE(SigLIP)** outperformed **LE(Label)** by a significant margin in our evaluations. We believe this can be attributed to a few causes: First, LE(SigLIP) can use a larger concept set, and as such can detect a wider range of concepts. Second it is likely more aligned with the simulator model, i.e. thinks of concepts in a more similar manner to the simulator. In contrast, LE(Label) utilizes some concepts that are very hard for a simulator to understand based on name alone, such as *leporid*, which is a superclass of rabbits. In section B.2.1 we tested using a simulator from a different model family, but SigLIP’s advantage still persisted. Regardless, it may be beneficial to use LE(Label) in some cases, for example to have more transparency and consistency in the explanations.

Computational Efficiency.

Both of our explanation methods take around 4 seconds per explained neuron on a machine with a single Tesla V100 GPU. In addition, **LE(SigLIP)** has a one-time cost of a single

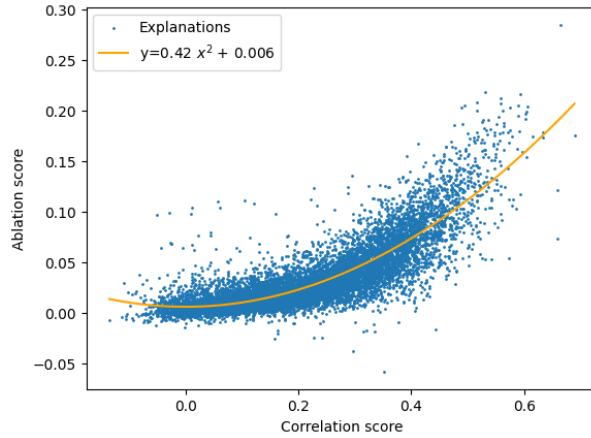


Figure 2.5. The relationship between correlation and ablation score of different explanations.

forward pass of $\mathcal{D}_{\text{probe}}$ through the SigLIP model which takes around 30 minutes. This does not need to be recalculated when explaining new neurons or models with the same data.

2.6 Related Work

2.6.1 Automated Interpretability in Vision

Several methods have been proposed to automatically explain the roles of individual neurons in neural networks. Network Dissection [6] is the first and likely most popular. They use a dataset with pixel-wise labeled concepts, and try to find concepts with high Intersection over Union (IoU) with binarized neuron activations. This method has a few downsides, such as treating neuron activations as binary and inability to detect concepts missing from their annotated dataset. Compositional Explanations [65] extends Network Dissection to deal with polysemantic neurons, that may activate on multiple unrelated concepts, by searching for logical compositions of concepts, i.e. a neuron could activate on Cat OR Dog. This is similar to our method in that they propose a way to compose a more complex explanation out of simpler primitive concepts. However we believe linear composition is preferable over logical composition because it naturally operates on scalar activations while logical composition requires binarizing neuron activations, and linear functions are much faster to search for than logical formulas are.

Recently [54] proposed an Extension to Compositional Explanations, addressing the problem that Compositional Explanations only explain the very highest activations. This was motivated by findings similar to our Section 2.2 showing that lower activation ranges are also important for network predictions. They propose dividing the activation range of a neuron into 5 buckets, and generating a separate compositional explanation for each. While this does provide a more complete picture of the neurons, the explanation is getting rather hard to understand, with 15 concepts per neuron that interact in complicated ways. We believe Linear Explanations are able to achieve a similar level of completeness of description with much less complexity.

Other relevant methods include MILAN [39] which trains a neural net to describe the most highly activating inputs of a neuron, DnD [4] which utilizes pre-trained models to produce generative explanations of highest activating inputs, [7] who propose doing Network Dissection with a Segmentation model instead of labeled data, and [16] who propose explaining neurons with Compositions of validation dataset labels similar to our LE(Label), but they use logical compositions instead of linear composition, and AUC to evaluate instead of ability to predict neuron activations.

Finally, recent papers CLIP-Dissect [68] and FALCON [46] have proposed methods that don't require labeled concept information at all by relying on supervision from multimodal models such as CLIP [77]. This is related to how our LE(SigLIP) works, but both previous methods only focus on explaining the very highest activating inputs.

2.6.2 Language Models Can Explain Neurons in LMs

[9] proposes an Automated Interpretability approach to explain individual neurons in large language models. They propose an explanation method similar to MILAN [39], and very different from ours, adapted to a language setting. They show a large model (GPT-4) the most highly activating inputs of a neuron and ask it to find what they have in common as the explanation for that neuron, which is studied in more detail by [55]. [9] also propose a new evaluation method: simulation, which evaluates explanations based on how well they can be used

to predict activations on new inputs, which we improve upon and extend to the vision setting in this work.

2.6.3 Unreliability of explaining only top activations

Recent works [66, 92] have shown that neuron explanations based on only highest activations are not robust, and can be easily manipulated. Similarly, feature visualizations can be easily fooled as shown by [29]. In light of this, our hope is that more holistic explanations explaining the entire activation range are less susceptible to such attacks.

2.6.4 Linear Probes

Linear Probing is a common method for finding explainable neurons in language models [3, 83, 37, 27]. In linear probing, a (sparse) linear model is trained to predict a concept based on neuron activations, with the goal to understand if/where the network represents that concept. Our method is effectively the *inverse* of linear probes, where we instead learn a sparse linear function of concepts to predict a neuron activation.

2.7 Conclusion

We have shown that only describing highly activating inputs as done by previous work is not sufficient to understand a neuron, and proposed a new solution called **Linear Explanations**. Our method explains neurons as a linear combination of concepts that produces highly accurate and complete neuron descriptions that are still easy to comprehend. Additionally, we developed a new way to more rigorously evaluate neuron explanations in vision models via simulation.

Impact statement

Our paper proposes improved methods to understand neural networks, and as such we expect its impact on society to be mostly positive. While our method is only a small part of this, we believe better understanding of networks reduces the chances of harm caused by deploying

unsafe or unreliable models in important settings. One potential downside of explanations is that they risk giving the illusion of understanding, without actually being faithful to the model. We hope to have reduced the chances of this with significant focus on better evaluations of neuron explanations.

Acknowledgements

This work is supported in part by National Science Foundation (NSF) awards CNS-1730158, ACI-1540112, ACI-1541349, OAC-1826967, OAC-2112167, CNS-2100237, CNS-2120019, the University of California Office of the President, and the University of California San Diego’s California Institute for Telecommunications and Information Technology/Qualcomm Institute. Thanks to CENIC for the 100Gbps networks. T. Oikarinen and T.-W. Weng are supported by National Science Foundation under Grant No. 2107189 and 2313105. T.-W. Weng also thanks the Hellman Fellowship for providing research support.

Chapter 2, in full, is based on the paper from Tuomas Oikarinen and Tsui-Wei Weng, ”Linear Explanations for Individual Neurons” published at the International Conference on Machine Learning (ICML), 2024. The dissertation author is the primary investigator and author of this paper.

Chapter 3

Evaluating Neuron Explanations: A Unified Framework with Sanity Checks

3.1 Introduction

Deep neural networks (DNNs) have achieved great success on a wide range of tasks, but they are very difficult to understand and often perceived as black-boxes [82]. To address this challenge, the field of mechanistic interpretability has recently emerged, aiming to provide a clearer understanding of the internal mechanisms of DNNs.

Providing natural language explanations for small components of a neural network is an important part of mechanistic interpretability. Classic work in this area includes Network Dissection [6] and other works explaining individual neurons in deep vision models [65, 39, 68, 4]. Other examples include automated neuron explanations [9, 55] for large language models, as well as explaining features of sparse autoencoders [14, 96].

Despite the introduction of various approaches for generating neuron explanations, existing papers often use very different metrics to evaluate how good their descriptions are, and it is not clear how they compare to each other. In addition, many evaluation metrics have problems, as shown by [42]. To ensure that unit explanations are reliable and trustworthy, it is crucial to establish a standardized framework for evaluation.

Motivated by the need for a standardized approach, in this work we unify many existing evaluation methods under a single mathematical framework: *NeuronEval*, which provides much

needed conceptual clarity to the topic of explanation evaluation. This framework allows us to clearly compare and contrast current evaluation techniques and provides a more transparent understanding of the evaluation pipeline. Inspired by this, we introduce two sanity tests to validate the metrics, revealing that most commonly used evaluation metrics fail at least one of these basic tests and should not be relied on alone. In summary, in this paper we:

- Formalize the task of evaluating individual unit explanations and unify 20 existing studies under the *NeuronEval* framework, which lets us apply standard statistical methods to evaluating neuron explanations.
- Propose two sanity checks for evaluation metrics: *Missing Labels Test* and *Extra Labels Test*. We then test different metrics both theoretically and empirically across hidden layer and final layer neurons, as well as on linear probes on both vision and language models.
- Out of the 18 metrics tested only the following pass the tests: *Correlation*(Pearson), *Cosine Similarity*, *AUPRC*, *IoU* and *F1-score*. In contrast, many commonly used evaluations such as: using biased *top-and-random* sampling, only evaluating on highly activating inputs (*Recall*), measuring *AUC* or mean activation difference (*MAD*), fail these tests and cannot be relied on by themselves.

Our code and results are available at https://github.com/Trustworthy-ML-Lab/Neuron_Eval.

3.2 Definitions

3.2.1 What is an individual unit in a neural network?

In this paper we are focused on **individual unit** explanations. By a unit, we mean *a smaller part of a neural network that can have an independent meaning*. The simplest form of such units is a single neuron, or a single channel of a Convolutional Neural Network (CNN), but a unit can be any scalar function of network inputs. Other interesting units that fit in our framework are linear combinations of neurons (i.e. directions in activation space), which are

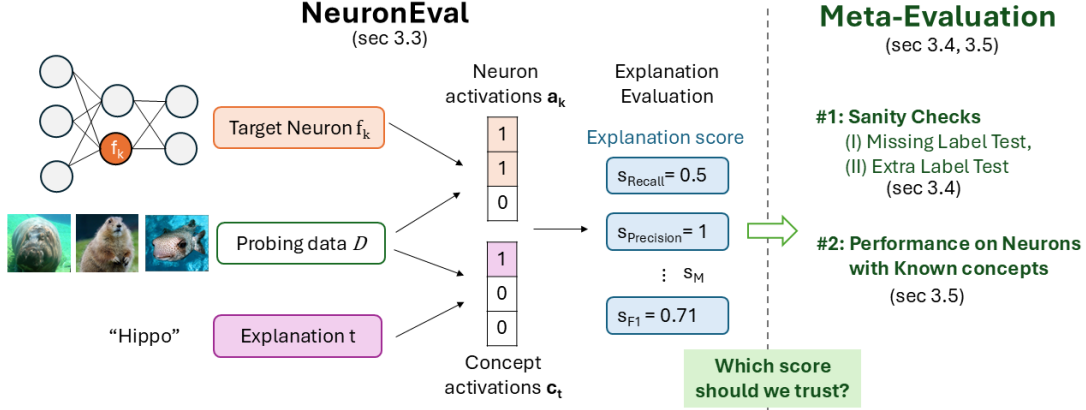


Figure 3.1. An overview of contributions. We first unify many existing explanation evaluation methods into a single mathematical framework *NeuronEval* containing 18 different metrics s_M . Next, we perform meta-evaluation via *NeuronEval* to answer the question: which evaluation metrics reliably measure how good an explanation is?

considered to correspond to a specific interpretable concept. These are used in studies such as TCAV [48], Concept Bottleneck Models [50, 108, 67, 93], Linear Probing [37] or steering vectors [94]. Finally, a unit could be a feature of a Sparse Autoencoder [18, 14] trained to disentangle a layer’s activations into interpretable individual components. In this paper, we focus on units with scalar activations, excluding larger components e.g. attention heads.

3.2.2 Problem Description

Our focus in this paper is explanation¹ *evaluation*. In evaluation, a text explanation t for a neuron k is given, and our goal is to evaluate how faithfully this explanation describes the neuron. Below we define evaluation as a function $\mathcal{E}$:

$$\text{Evaluation (of Explanation } t) : \mathcal{E}(f_k^{0:l}, \mathcal{D}, t) \rightarrow \mathbb{R}$$

Here $\mathcal{E}$ is a function that takes a probing dataset $\mathcal{D}$, a neural network² f and a text description t , and returns a scalar score, where a higher score indicates the description is better

¹Note: Throughout the paper we use ”explanation”, ”concept”, and ”description” interchangeably to refer to a text description.

²We can write neuron k in layer l of neural network f as a function $f_k^{0:l}(x)$, where $f^{i:j}$ represents the i through j ’th layers of the neural network $f(x)$. Here we assume the units are neurons for notational simplicity.

i.e. more reliable/faithful. See Figure 3.1 for an illustration of the evaluation process.

For explanations to be reliable, evaluating explanations is very important, but the community has not reached an agreement on which $\mathcal{E}$ to use. Currently different papers often use different $\mathcal{E}$ without theoretical justifications. Thus, one focus of our paper is to perform *meta-evaluation* and rigorously investigate different evaluation metrics $\mathcal{E}$.

3.2.3 What are the goals of an explanation?

Neuron explanations are typically generated to improve our understanding of the model, which can then help improve, for example, safety and reliability of the models. However, this goal is vague and hard to measure generally. We believe a more precise definition of the goals of an explanation is essential for thinking clearly about how to evaluate them.

In this paper, we focus on **Input-based** neuron explanations, which make up the majority of existing explanations. We argue that the goal of an explanation is to provide a human-interpretable approximation of the following function:

$$\textbf{Input} \rightarrow \textbf{Unit Activation: } x \rightarrow f_k^{0:l}(x).$$

A good explanation should be able to describe which inputs cause a high unit activation, and which do not. Note that there are other type of explanations being proposed (e.g. output-based, see App. C.2.2 for discussion.), which is not the focus of this work.

3.3 A Unified Evaluation Framework

We start by studying some popular neuron explanation evaluations in Sec. 3.3.1 and showing they can be formalized under the same notation. We then take a further step in Sec 3.3.2 to show that almost all existing work can be unified under this evaluation framework, which we call *NeuronEval*.

3.3.1 Formalizing Example Evaluations

Example 1: Crowdsourced Highly Activating Inputs Evaluation.

Currently, perhaps the most common way to evaluate neuron explanations is to show (crowdsourced) human raters examples of highly activating inputs, and ask the raters whether these inputs can be described by the explanation (also known as ‘concept’) [114, 6, 68].

To analyze this approach more rigorously, we describe it through the following mathematical formulation: Given a set of probing data $\mathcal{D} = \{x_i\}$, we first define the **neuron activation vector** a_k for neuron k as:

$$a_k \in \mathbb{R}^{|\mathcal{D}|}, [a_k]_i = f_k^{0:l}(x_i) \quad \forall i \in [1, |\mathcal{D}|]. \quad (3.1)$$

Here, the i -th entry of a_k denotes the neuron activation on i -th input x_i . To represent highly activating inputs, we define a binarization function $B : \mathbb{R}^n \rightarrow \{0, 1\}^n$, that takes a real vector and turns its elements into 1 if it is a “high” activation and 0 otherwise. In particular, we use the top_α binarization function for neuron activations, where the activations in top α percentile return 1, and the rest return 0 as this is how highly activating inputs are typically selected. See Appendix C.1.1 for a more detailed description.

Next, we describe the human ratings for the presence of text concept t in the inputs by defining the **concept activation vector** c_t :

$$c_t \in \mathbb{R}^{|\mathcal{D}|}, [c_t]_i = \mathbb{P}(t|x_i) \quad \forall i \in [1, |\mathcal{D}|] \quad (3.2)$$

Here each element of c_t , i.e. $[c_t]_i$, represents the fraction of raters that agreed on concept t being present on that input x_i . Finally, given a_k and c_t , we denote s as an explanation score to quantify whether neuron k is well explained by the concept t . In this evaluation, the fraction of highly activating inputs that have concept t present is used to represent the quality of explanations. To formalize this we can define s in terms of a_k and c_t as follows:

$$s := \mathbb{P}(\text{concept} | \text{neuron is active}) = \frac{B(a_k) \cdot B(c_t)}{\|B(a_k)\|_1} \quad (3.3)$$

since the number of highly activated inputs can be computed as $\|B(a_k)\|_1$ and the number of highly activated inputs that have concept t present is $B(a_k) \cdot B(c_t)$. Note the value of c_t on not highly activating inputs doesn't affect the score so we don't need to evaluate it on all inputs. Here we also binarize the concept vector c_t by rounding to 1 for elements ≥ 0.5 (indicating concept probability ≥ 0.5) and to 0 otherwise.

If we consider the goal of an explanation to be predicting neuron activations given whether the concept is present, i.e. *simulation*, then we can formulate the neuron explanation as a binary classification problem. Specifically, as we show in Appendix C.1.2, equation 3.3 is actually equivalent to measuring "Recall":

$$s_{\text{Recall}} = \frac{TP}{TP + FN} = \frac{B(a_k) \cdot B(c_t)}{\|B(a_k)\|_1} \quad (3.4)$$

So crowdsourced evaluation of highly activating inputs is equivalent to only measuring "Recall", i.e. we have $s = s_M$ with metric $M = \text{Recall}$.

Example 2: IoU with Labeled Data.

In addition to crowdsourcing, other types of evaluations have been used for validating neuron explanations. For example, [6] collected a curated labeled dataset to evaluate neuron explanations by measuring the *Intersection over Union* (IoU, also known as Jaccard index) between neuron activations and concept labels on the annotated dataset. This measures $\mathbb{P}(\text{concept AND neuron is active}) / \mathbb{P}(\text{concept OR neuron is active})$.

We can also describe this evaluation with the same formalism as in **Example 1**, but *with two main changes*. First, we use a different *concept source*, where the elements of c_t come from the labeled data instead of crowdsourced ratings. Second, we set the metric M as IoU, giving us the following evaluation score:

Table 3.1. Summary of **evaluations** used in existing work. $\sim$ indicates using a metric with small differences from our definition, while T&R indicates use of biased *top-and-random* sampling to evaluate the metric with fewer samples. See Table C.3 for an extended version.

<i>NeuronEval</i> : A General Framework to Evaluate Neuron Explanations				
Metric M	Study	Concept Source c_t	Granularity	Domain
Recall	[114]	Crowdsourced	Whole Input	Vision
$\sim$ Recall	[6, 68] [67, 4]	Crowdsourced	Whole Input	Vision
Precision	[91]	Generative	Whole Input	Vision
F1-score	[42] [37]	Generative + Model Labeled data	Whole Input Per-token	Language Language
IoU	[6, 65] [54]	Labeled data	Per-pixel	Vision
Accuracy	[50]	Labeled data	Whole Input	Vision
$\sim$ AUC	[117]	Crowdsourced	Whole Input	Vision
Inverse AUC	[16] [51]	Labeled data Generative	Whole Input Whole Input	Vision Vision
Correlation(T&R)	[9]	Model	Per-token	Language
Correlation	[69]	Model	Whole Input	Vision
Spearman Correlation(T&R)	[14, 96]	Model	Per-token	Language
$\sim$ WPMI	[68]	Model	Whole Input	Vision
MAD	[51]	Generative	Whole Input	Vision
$\sim$ MAD	[86] [88]	Generative Generative	Whole Input Whole Input	Vision Language

$$s_{\text{IoU}} = \frac{B(a_k) \cdot B(c_t)}{\|B(a_k)\|_1 + \|B(c_t)\|_1 - B(a_k) \cdot B(c_t)} \quad (3.5)$$

Example 3: Automated Simulation Evaluation with Language Models.

More recently, [9] evaluate neuron explanations of large language models by measuring the correlation between actual neuron activations and activations predicted by a language model given the explanation.

While in different domains (Examples 1 and 2 are from vision models), this evaluation can also be formalized in the same manner as Example 1 and 2. First, the *concept source* for c_t is now model predicted neuron activations scaled 0-1, as these are essentially pseudo-labels for

the presence of the concept in the given input. Second, for the metric M , we now set it to be the Pearson’s Correlation coefficient and obtain the evaluation score $s_{\text{Correlation}}$ as:

$$s_{\text{Correlation}} = \frac{1}{|\mathcal{D}|} \frac{\sum_i ([a_k]_i - \mu(a_k)) \cdot ([c_t]_i - \mu(c_t))}{\sigma(a_k) \sigma(c_t)} \quad (3.6)$$

where $\mu(z)$ and $\sigma(z)$ is the mean and standard deviation of the vector z .

3.3.2 Unifying Neuron Explanation Evaluation: *NeuronEval*

As we have shown in the previous three examples (Equations 3.4, 3.5 and 3.6), evaluations that look extremely different on the surface, can actually be written as simple functions of the neuron activation vector a_k and concept activation vector c_t . This means the explanation score s_M can be written as a function $s_M(a_k, c_t)$ of a_k and c_t , that is, $\mathcal{E}(\mathcal{D}, f_k^{0:l}, t) = s_M(a_k, c_t)$, where M is the name of the specific metric chosen. For convenience, we call this the *NeuronEval* Framework, showcased in Figure 3.1. Going further, we find that, as we show in Table 3.1, almost all existing methods in the literature for **evaluation** $\mathcal{E}$ can be formalized under this framework.

Notably, *NeuronEval* is a general framework – as shown in Table 3.1, it includes not only the standard statistical metrics (e.g. Recall, F1-score, Correlation), but also other metrics such as Mean Activation Difference (MAD), which compares the average neuron activation on inputs where the concept is present vs those where concept is missing:

$$s_{\text{MAD}} = \frac{B(c_t) \cdot a_k}{\|B(c_t)\|_1} - \frac{(\mathbf{1} - B(c_t)) \cdot a_k}{\|\mathbf{1} - B(c_t)\|_1}. \quad (3.7)$$

Other standard metrics include Precision [91], which can be intuitively understood as measuring $\mathbb{P}(\text{neuron is active} | \text{concept})$, defined as:

$$s_{\text{Precision}} = \frac{B(a_k) \cdot B(c_t)}{\|B(c_t)\|_1} \quad (3.8)$$

Formalizing metrics under the same framework also highlights gaps in existing work, leading us to test standard statistical metrics that were not explored by previous work, such as AUPRC (defined in Appendix C.1.3).

Overall we have aimed to include all evaluation metrics used by previous works, as well as standard statistical metrics not previously explored, for a total of 18 different metrics and evaluations from 20 studies within the *NeuronEval* framework, which we compare in our experiments (Section 3.4 and 3.5), and the framework can easily include more in the future. In Appendix C.1.3 we define all the other metrics evaluated, such as F1-score, Cosine similarity, AUC and AUPRC, as well as additional details on these metrics.

Existing Work as Special cases of *NeuronEval*.

We summarize how most existing evaluations fit into our *NeuronEval* framework in Table 3.1 (see Appendix C.2.2 for discussion on a few exceptions). This includes not only evaluations of individual neuron explanations, but also explanations of SAE features [14], CBM neurons [50], or linear probes [37]. Writing diverse evaluations as special cases of the same framework allows us to more clearly think about each component of an evaluation in isolation, and can help develop more effective evaluations by combining the best parts of previous studies. We classify differences in existing evaluations into four main components:

- (i) **Evaluation metric M :** This is the main focus of our paper, to analyze which evaluation metrics are good choices.
- (ii) **Source of Concept Vector c_t :** There are many choices for the concept vector c_t . These include, but are not limited to: labels from a labeled dataset, using a model to create pseudo-labels, using a human evaluator, or generating new inputs and using the prompts as labels.
- (iii) **Granularity of activation vectors:** The simplest case is full input level activations, i.e. $|\mathcal{D}| = |a_k| = |c_t| = n$. These can also be more specific, for example pixel-level activations

as is the case in Network Dissection [7], or token level as is often the case for language model explanations.

- (iv) **Probing dataset $\mathcal{D}$:** Which inputs are used for the evaluation. Typical choices include the training/validation data of the model, a special labeled dataset designed for probing, or different datasets for different concepts. Importantly, the dataset used for evaluation should be disjoint from the dataset used for explanation generation to avoid overfitting.

Importantly, regardless of the answer to these questions, or the application domain (e.g. vision or language), each evaluation fits under the same mathematical formalism of *NeuronEval*.

3.4 Meta-Evaluation #1: Sanity Checks for Evaluation Metrics

Now that we have organized previous evaluations under the *NeuronEval* framework, we can isolate and perform *meta-evaluation* on one particular component of the evaluation: the choice of metric M . As seen in Table 3.1, existing work uses a wide variety of metrics, often without justification. In order to understand which evaluation metrics are reliable, we propose two sanity checks: Missing Labels Test and Extra Labels Test which we describe in detail in this section. These tests are inspired by [2], and passing them is intended to be a necessary (not sufficient) condition for a reliable evaluation metric.

3.4.1 A Motivating Example

We start by analyzing a simple failure case of the precision and recall metrics. Let the probing dataset contain 6 images of animals x_i ordered as $\{\text{dog, cat, dog, bear, monkey, flamingo}\}$. Suppose we have a "pets" neuron that only activates on pets (i.e. dogs or cats), then we can compute the neuron activation vector a_k as $B(a_k) = [1, 1, 1, 0, 0, 0]^\top$. We can also compute the concept activation vector c_t for different concepts t based on whether the input image x_i contain the concepts, giving us: $c_{\text{dog}} = [1, 0, 1, 0, 0, 0]^\top$, $c_{\text{cat}} = [0, 1, 0, 0, 0, 0]^\top$, $c_{\text{pet}} = [1, 1, 1, 0, 0, 0]^\top$,

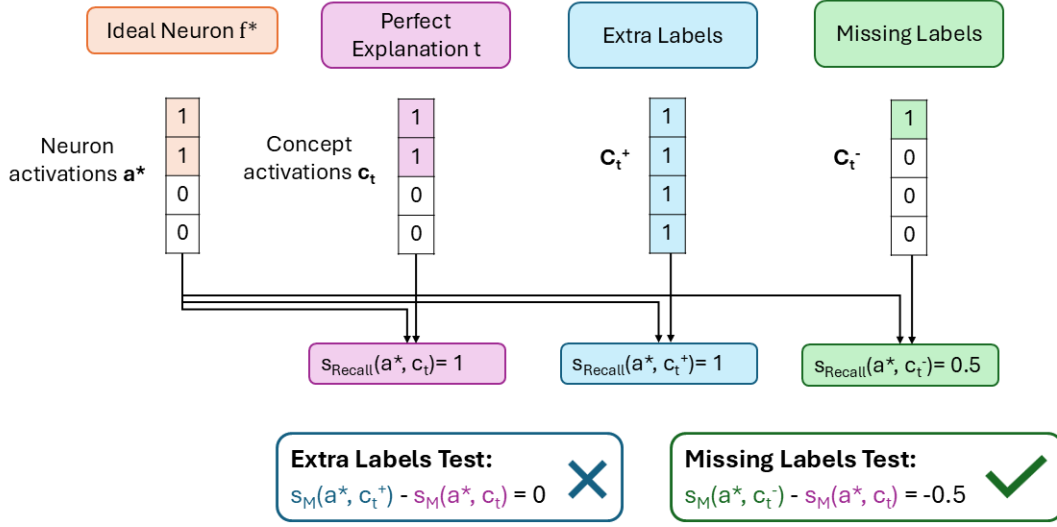


Figure 3.2. Overview of our theoretical Missing and Extra Labels Tests. We can see the Recall metric fails the extra labels test as it cannot differentiate between the perfect explanation and an explanation with extra labels. On the other hand Recall clearly passes the missing labels test as $-0.5 \ll -\epsilon = -0.001$

$\mathbf{c}_{\text{animal}} = [1, 1, 1, 1, 1]^\top$. We can then calculate the neuron explanation score s_M of this "pets" neuron with different metrics M in the following table:

Table 3.2. A 'pet' neuron: different metrics M gives different evaluation scores s_M on explanations t . For the metrics $M = \{\text{Recall}, \text{Precision}, \text{IoU}\}$, 1 is the perfect score, and 0 is the worst score.

Explanation t	s_{Recall}	$s_{\text{Precision}}$	s_{IoU}
<i>dog</i>	0.67	1	0.67
<i>cat</i>	0.33	1	0.33
<i>pet</i> (ground truth)	1	1	1
<i>animal</i>	1	0.5	0.5

When comparing different evaluation metrics, we can see that *Recall* cannot distinguish between the correct concept (pet) and a concept that is too generic (animal), as s_{recall} gives perfect score for both explanations. On the other hand, *Precision* favors concepts that are too specific (dog, cat), as $s_{\text{precision}} = 1$ for 'dog', 'cat' and 'pet'. In contrast, IoU can unambiguously determine the correct concept 'pet', as $s_{\text{IoU}} = 1$ only for 'pet'. See Figure C.1 for illustration.

3.4.2 Sanity Test Definitions

Inspired by this example, we propose two general tests to measure whether a certain metric is too biased towards specific or generic concepts. These tests are motivated by the following idea: *For a reliable metric, a correct description for a neuron should get the highest score, while its subset (too specific) or superset (too general) should score lower.*

(I): Missing Labels Test

The missing labels test is a generalized test to measure whether a metric can differentiate between the correct concept c_t and a concept c_t^- that is too specific (a random subset of the correct concept). In particular we generate c_t^- by randomly replacing half of the elements of c_t with 0. That is we remove half of the concept labels for concept t . $\mathbb{E}[\|c_t^-\|_1] = \|c_t\|_1/2$:

$$[c_t^-]_i = \begin{cases} [c_t]_i & \text{with probability 0.5} \\ 0 & \text{with probability 0.5} \end{cases} \quad (3.9)$$

We then measure the score difference Δs , i.e. how much does the evaluation score change when replacing the original concept vector c_{t_k} with a concept that is too specific, c_t^- :

$$\Delta s(k) = \mathbb{E}_{c_{t_k}^\pm} [s_M(a_k, c_{t_k}^\pm) - s_M(a_k, c_{t_k})] \quad (3.10)$$

For the sanity test, it is not important how much the modified labels decrease the score. We simply want it to decrease score by a non-zero amount on all neurons, which leads us to define our main metric: *Decrease Acc*.

$$\text{Decrease Acc} = \frac{1}{|K|} \sum_{k \in K} \mathbb{1}[\Delta s(k) < -\varepsilon] \quad (3.11)$$

Here K is the set of neurons looked at and t_k is the best/correct concept for neuron k . Note that we normalized the scores such that maximum of M is 1 and minimum value is 0 to allow for equal comparison between metrics. We set $\varepsilon = 0.001$ in our experiments to require

a small but noticeable change in scores. We study the sensitivity of these tests to the choice of ε further in Appendix C.5.4. The assumption behind this test is that if concept t_k is a good description for neuron k , a random subset of it should be a worse description.

Table 3.3. Averaged experimental and theoretical results of our missing labels and extra labels test. Experimental results are averaged across settings like vision and language domain, while theoretical results are averaged over different neuron activation frequencies. (T&R) indicates the use of top-and-random sampling. We can see most evaluation metrics fail at least one of the tests.

Meta-Evaluation #1	(I) Missing Labels Test		(II) Extra Labels Test		Pass
	Experimental	Theoretical	Experimental	Theoretical	
Recall	98.66%	100.00%	0.00%	0.00%	×
Precision	45.73%	0.00%	99.81%	100.00%	×
F1-score	93.68%	100.00%	99.82%	100.00%	✓
IoU	93.62%	100.00%	99.81%	100.00%	✓
Accuracy	23.79%	60.00%	70.37%	69.68%	×
Balanced Accuracy	98.65%	100.00%	53.67%	60.00%	×
Inverse Balanced Accuracy	64.18%	60.00%	99.50%	100.00%	×
AUC	94.96%	100.00%	59.18%	60.00%	×
Inverse AUC	52.81%	60.00%	99.99%	100.00%	×
Correlation	99.41%	100.00%	99.92%	100.00%	✓
Correlation (T&R)	87.83%	100.00%	60.26%	43.64%	×
Spearman Correlation	64.05%	67.20%	49.21%	44.08%	×
Spearman Correlation (T&R)	80.04%	100.00%	59.81%	19.68%	×
Cosine	99.45%	100.00%	99.26%	100.00%	✓
WPMI	95.89%	100.00%	58.84%	100.00%	×
MAD	59.81%	60.00%	99.34%	100.00%	×
AUPRC	95.61%	100.00%	99.46%	100.00%	✓
Inverse AUPRC	99.15 %	100.00%	95.58%	89.54%	×

(II): Extra Labels Test

The extra labels test is essentially the opposite of missing labels test, where we test whether a metric can differentiate between a correct concept and a concept that is too generic i.e. a random superset of the concept. In particular, we create c_t^+ by randomly doubling the size of c_t , i.e. $\mathbb{E}[\|c_t^+\|_1] = 2\|c_t\|_1$. That is:

$$[c_t^+]_i = \begin{cases} 1 & \text{if } [c_t]_i = 1, \text{ else with probability } \frac{\|c_t\|_1}{n - \|c_t\|_1} \\ 0 & \text{otherwise} \end{cases} \quad (3.12)$$

where n is the length of vector c_t . Similar to the Missing labels test, to pass the Extra Labels test, a metric should have a high Decrease Acc as defined in equation 3.11. If concept t is a good description for neuron k , giving additional positive concept labels to random inputs should decrease its similarity score. For simplicity, we only apply these tests with ground truth labels as concept source where c_t is binary.

3.4.3 Test Setup

We perform two versions of these tests, **Experimental** on real neurons across diverse settings, and **Theoretical** on *ideal* neurons described below.

Experimental: We experimentally evaluated these metrics on neurons from 8 different settings across vision and language models, covering final layer neurons, hidden layer neurons, CBM neurons and linear probe outputs. We evaluated vision models across 3 datasets: Imagenet, Places365 and CUB200, while language models were evaluated on a subset of OpenWebText[32]. See Appendix C.6 for detailed description of the evaluation setting and results on individual datasets. We report the Averaged results in Table 3.3.

Theoretical: In addition, we perform theoretical analysis on the Missing/Extra labels test for hypothetical neurons whose activations perfectly match a concept, which we discuss in detail in Appendix C.3. See Figure 3.2 for an overview. In this setting, the missing labels test measures whether a metric can differentiate a concept that perfectly matches neuron activations c_t (Recall=1, Precision=1) from a concept c_t^- (with Recall=0.5 and Precision=1). Similarly, the Extra Labels test measures differentiation between the perfect concept and c_t^+ (with Recall=1 and Precision=0.5). We simulate neurons with many different activation frequencies, and averaged results are reported in Table 3.3.

Finally, we analytically studied the expected change in scores for different binary metrics under Missing and Extra labels test in Appendix C.4. Our analytical results are consistent with the empirical findings in Table 3.3 and Fig. 3.3.

3.4.4 Results and Discussion

Results:

In Table 3.3 we report the averaged results of this test across these two sets of neurons for all different evaluation metrics. For simplicity, we say a metric passes if its Decrease Acc $> 90\%$ for all tests. Failing methods are marked in red color. Overall, our theoretical results closely match our empirical observations. Based on test results, we can group the metrics as follows:

- **Fail Both:** *Accuracy* and *Spearman Correlation* perform poorly in both tests as their score is largely determined by the majority of inputs that neither activate the neuron nor have the concept.
- **Fail Missing Labels Test:** *Precision*, *Inverse Balanced Accuracy*, *Inverse AUC* and *MAD* are biased towards too specific concepts.
- **Fail Extra Labels Test:** *Recall*, *Balanced Accuracy*, *AUC* and *Correlation(T&R)* are biased towards concepts that are too generic.
- **Borderline:** Our results are inconclusive for the following two metrics: *WPMI*, which passes the tests on certain hyperparameter choices but not on others, and *Inverse AUPRC*, which fails the theoretical test only on perfectly balanced data but passes the empirical tests.
- **Passing:** *F1-score*, *IoU*, *Correlation*, *Cosine similarity* and *AUPRC* perform well and pass the tests.

Importantly, we think any metric failing these tests should not be relied on by itself, and encourage the use of passing metrics. We also perform an ablation in Appendix C.5.1 showing that our findings are not sensitive to specific parameter choices of the test.

Concept Imbalance Explains Test Failures:

In our theoretical experiments, we discover that the main reason for a metric failing one of these tests is poor handling of concept imbalance. Figure 3.3 shows an example of this

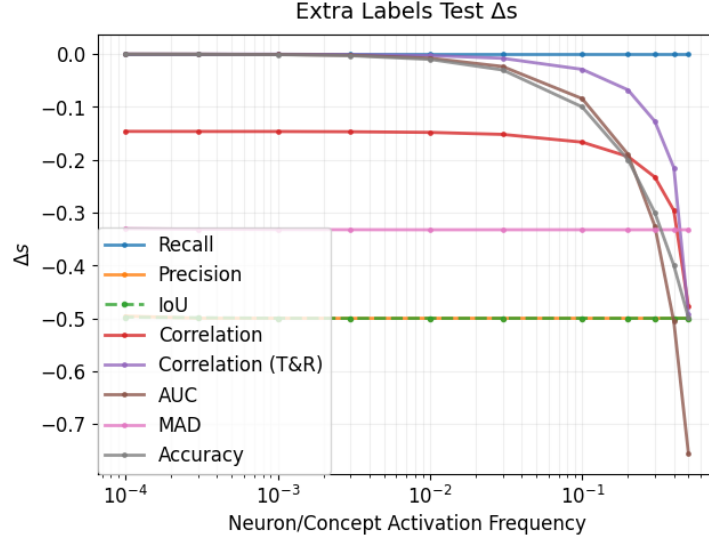


Figure 3.3. The effect of Concept/Neuron Activation frequency on the score difference in Theoretical Extra Labels test. We can see that all the metrics that fail the Extra Labels test approach 0 score difference on neurons/concepts that rarely activate, i.e. where the concept labels are imbalanced. The passing metrics maintain a non-zero score difference regardless of how rare the concept is.

phenomenon. On balanced neurons/concepts that activate on more than 10% of the inputs, all metrics except for Recall pass the Extra Labels Test (large $|\Delta s|$). However, on neurons that activate less frequently, the failing metrics such as Accuracy and AUC can no longer distinguish between the perfect concept and an overly generic concept, and the score difference approaches 0. Therefore, failing the Missing/Extra Labels test can in most cases be attributed to the metrics being unable to handle imbalanced data. In this light, our results align with previous statistical knowledge, and metrics such as Accuracy and AUC which are not great for imbalanced datasets fail the tests, while metrics designed for imbalanced data such as F1-score and AUPRC pass the tests. Since real units in neural network often activate on a wide range of frequencies, and are sometimes encouraged to activate very rarely (i.e. Sparse Autoencoders), it is essential to use evaluation metrics that can handle imbalanced data.

Top-and-random sampling: Following [9, 14], we also tested a version of Correlation and Spearman Correlation that uses top-and-random sampling (T&R), where 50% of the evaluated samples are highly activating inputs only and the remaining 50% are sampled uniformly at

random. For our T&R experiments we sampled 25 inputs from the top 0.2% highest activating inputs and 25 random inputs. While *Correlation* passes both tests with uniform sampling, top-and-random sampling makes it fail the extra labels test. This is not surprising, as greatly oversampling highly activating inputs makes the metric more similar to Recall that only evaluates on highly activating inputs. This also explains why [42] found explanations from [9] with very high Correlation(T&R) scores to have relatively low F1-scores.

c_t from Generative models: We discover that evaluations based on generating new inputs with generative models are limited to running evaluations that fail the missing labels test and are biased towards more specific concepts. This is because c_t based on generative model prompts is inherently incomplete or *missing labels*, and as such should not be solely relied on. See Appendix C.2.1 for further discussion on this situation and potential remedies.

3.5 Meta-Evaluation #2: Performance on Neurons with Known Concepts

In this section we perform an additional comparison between evaluation metrics by empirically comparing how well they perform on neurons where we know their *ground truth* function, such as neurons in the final (classification) layer of a model.

3.5.1 Method: AUPRC between scores of correct and incorrect explanations

The idea of this test is to directly measure how good specific evaluation metrics are. To do this, we have a concept set C that includes the correct explanation for each neuron $k \in K$, as well as some incorrect explanations. K is the set of neurons evaluated such as all neurons in a layer. We then score all $|K||C|$ (neuron, explanation) pairs and measure whether the metric consistently assigns higher scores to pairs with correct explanations over incorrect ones.

In particular, we measure the AUPRC (area under precision-recall curve) since this is a very imbalanced task with most (neuron, explanation) pairs being incorrect. Mathematically,

Table 3.4. Comparison of different evaluation metrics on neurons with known concepts, averaged across 8 settings. Lower rank means better performance. Best performing metric in **bold**, and second best underlined.

Method	Avg. AUPRC	Avg. Rank
Recall	0.6722	11.30
Precision	0.8039	7.90
F1-score/IoU	0.8140	6.70
Accuracy	0.7215	10.80
Balanced Accuracy	0.7979	7.30
Inverse Balanced Acc.	0.8087	7.10
AUC	0.7652	11.00
Inverse AUC	0.7569	10.90
Correlation	0.8765	1.60
Correlation(T&R)	0.6606	10.70
Spearman Correlation	0.0853	16.20
Spearman Correlation(T&R)	0.3418	15.40
<u>Cosine</u>	<u>0.8666</u>	<u>2.30</u>
WPMI	0.7999	7.30
MAD	0.6952	8.80
AUPRC	0.8406	3.90
Inverse AUPRC	0.6904	9.30

we can write the predictions $\hat{Y}_M$ as: $\hat{Y}_M = \{s_M(a_k, c_t) \mid \forall k \in K, t \in C\}$, while the labels Y are $Y = \{\mathbb{1}[t = t_k^*] \mid \forall k \in K, t \in C\}$, where t_k^* is the correct explanation for that neuron. Our meta-AUPRC evaluation can then be defined as:

$$\text{meta-AUPRC}(M) = \text{AUPRC}(\hat{Y}_M, Y) \quad (3.13)$$

A metric reaches perfect meta-AUPRC of 1 if all correct (neuron, explanation) pairs get a higher score than all incorrect pairs, while random guess performance is ~ 0 .

Experimental Setup.

Similar to section 3.4, we ran these tests on 10 different setups, consisting of 5 separate models, 4 datasets, vision and language domains and both ground truth labels and pseudo-labels as concept source for c_t . See Appendix C.6 for detailed description of experimental setup and details on individual models.

For all experiments we split a random 5% of the neurons into validation set. For metrics that require hyperparameters such as α , we use the hyperparameters that performed the best in terms of Meta-AUPRC on the validation split for each setting. We then report performance on the remaining 95% of neurons. In Table 3.4, we report the average scores and average ranks (i.e. the best metric for each setup gets 1, the worst gets 17) of the metrics across the four setups for both tasks. See Appendix C.6 for detailed breakdown of results.

We can see that in general the metrics that passed our tests in Section 3.4 perform better than those that didn't, with the top performance going to Correlation, followed by Cosine similarity then AUPRC and F1-score/IoU³. Overall we notice continuous metrics perform somewhat better than binary ones. This is likely because binarizing neuron activations loses valuable information, and it is hard to find one binarization threshold α that works well for all neurons. Despite good performance here, the *Cosine* metric is very sensitive to the average activation of a neuron as we show in C.5.2, which raises some concerns regarding relying on it.

3.6 Discussion

3.6.1 Relevance of our tests to Realistic Failure modes

While our sanity test definition in Section 3.4 is relatively abstract, we think that most real world explanation failures are closely connected to one of two failure modes, and that these failure modes are directly captured by our tests:

- **FM-A) Explanation too generic:** This means the explanation concept is a superset of the “true” neuron concept, e.g. describing a neuron as “animals” when it only activates on dogs. Our Extra Labels Test captures whether a metric can detect this failure mode.
- **FM-B) Explanation too specific:** Explanation concept is a subset of real neuron activations, i.e. describing a neuron as “black cat” when it activates on all cats. Our Missing

³Note we combined F1-score and IoU into same element in this table as they always return the same ordering between pairs leading to identical AUPRC, see Appendix C.1.4 for proof.

Labels Test captures whether a metric can detect this failure mode.

Since our idea of a “concept” is very general, it can include any text-based description. This means a single concept could be highly specific (e.g. “flying bird”), or a composition of simpler concepts (e.g. “water OR river”). Below we show how common explanation failures relate to our tests.

- **Polysemanticity:** A popular model of polysemanticity is to model neuron activations as an OR of different concepts. If the explanation only captures one of these concepts, this means the explanation is too specific (FM-B).
- **Context specific activations:** A context specific neuron activation such as *flying bird* means the neuron’s “true” concept is a subset of the non-context specific concept i.e. *bird*. If the explanation is not context-specific, the explanation is too generic (FM-A).
- **Non activating inputs still contain the concept:** This is a common explanation failure case also known as spurious correlations, where all the highly activating inputs share a concept, but not all inputs with this concept make the neuron activate. This means that the explanation concept is too generic i.e. a superset of the “true” concept (FM-A).

Finally, in Appendix C.5.3 we conducted a test using real semantic superclasses of the concept as c^+ instead of our random superset, and found that the results are essentially identical, highlighting the relevance of our tests to real world settings.

3.6.2 Conclusions

In this paper, we have created the *NeuronEval* framework for unifying different evaluations under the same mathematical formalism, clarifying the definitions around evaluating unit explanations. We have also proposed new sanity tests which lead to our main finding: **Neuron Explanation Evaluations should use metrics that pass the Missing and Extra Labels Tests.** We identified Correlation, Cosine, AUPRC, F1-score and IoU as solid passing metrics in our

study of 18 different metrics. Popular evaluation metrics used in the literature such as Recall and AUC as well as using top-and-random sampling do not pass these tests, and should not be relied on by themselves. With evaluation on neurons with ground truth explanation available (e.g. final layer neuron), we identified that the top-performing metrics overall are Correlation, Cosine and AUPRC. The final choice of metric may be further affected by considerations such as labeling cost discussed in C.2.2 or other failure modes such as Cosine’s sensitivity to average activation showcased in C.5.2.

Impact Statement

Our paper is focused on rigorous evaluation of neural network interpretability results, and as such we believe it will have a largely positive societal impact. Being able to understand/see inside models is likely to be very useful in deploying models in a reliable and safe manner. In addition, rigorously evaluating their interpretations is very important so we can avoid relying on unfaithful explanations or interpretability illusions and can place appropriate trust in these models.

Acknowledgements

The authors are partially supported by National Science Foundation under Grant No. 2107189, 2313105, 2430539, Hellman Fellowship, Intel Rising Star Faculty Award, Nvidia Academic Grant Program as well as the U.S. National Science Foundation Institute for Learning-enabled Optimization at Scale (TILOS; NSF CCF-2112665). The authors would like to thank anonymous reviewers for valuable feedback to improve the manuscript.

Chapter 3, in full, is based on the paper from Tuomas Oikarinen, Ge Yan and Tsui-Wei Weng, ”Evaluating Neuron Explanations: A Unified Framework with Sanity Checks” published at the International Conference on Machine Learning (ICML), 2025. The dissertation author is the primary investigator and author of this paper.

Chapter 4

Beyond Top Activations: Efficient and Reliable Crowdsourced Evaluation of Automated Interpretability

4.1 Introduction

Despite their transformative capabilities, deep learning models remain fundamentally opaque, which limits their reliability and trustworthiness, especially in high-stakes domains. To address this, the field of mechanistic interpretability, which aims to understand the mechanisms inside neural networks has grown rapidly. A key part of mechanistic interpretability is understanding small components of neural networks such as individual neurons or latents in a sparse autoencoder (SAE) through text-based descriptions, which can be achieved via automated neuron descriptions [7, 65, 39, 68, 69, 16, 54, 86, 4].

To assess the quality of these neuron explanations, researchers often rely on human studies, such as crowd-sourced evaluations via platforms like Amazon Mechanical Turk (AMT). However, despite their importance, the design and analysis of these studies has received relatively little attention. Most existing evaluations examine only whether an explanation matches the top activating inputs of a neuron. As identified by [71], this corresponds to measuring only *Recall*, and ignores many important factors such as whether the explanation also describes lower activations of that neuron, or whether all inputs matching the explanation actually activate the neuron (*Precision*). This can lead to misleading conclusions and fails to reliably assess

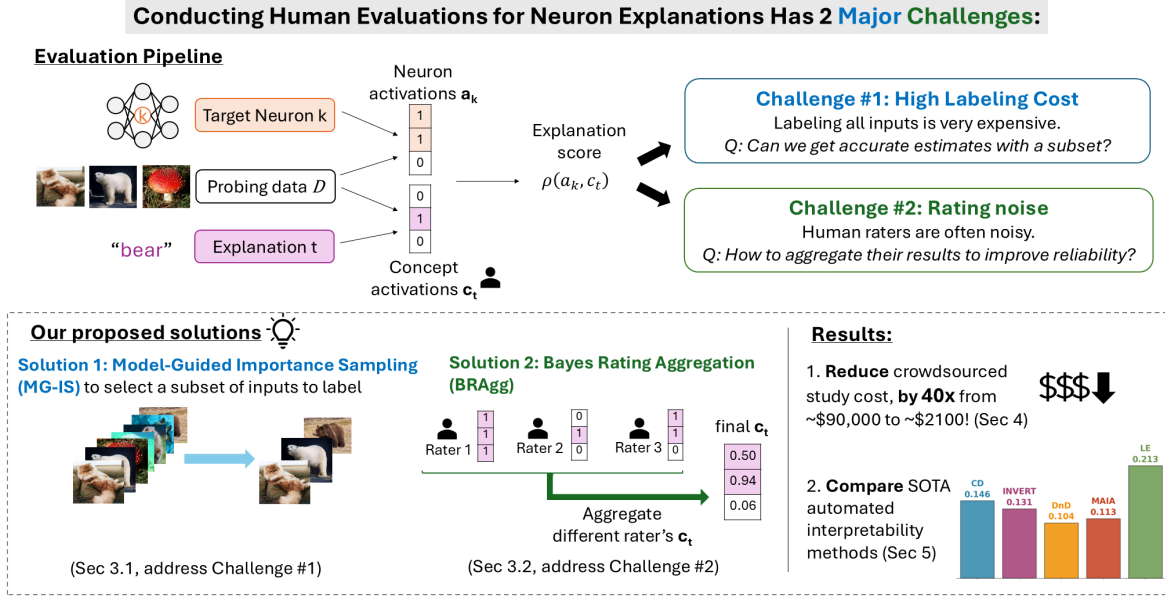


Figure 4.1. Overview of the explanation evaluation pipeline. We focus on two main challenges: 1: How to reduce high labeling cost? and 2: How to effectively handle noisy ratings? Our proposed solutions are discussed in Section 4.3, validated in Section 4.4, and applied to compare automated interpretability methods in Section 4.5.

explanation quality.

Motivated by the above limitations, we aim to conduct a crowdsourced evaluation using a more principled evaluation metric: the *correlation coefficient* between neuron activations and concept (explanation) presence labels. We choose correlation as it was identified as the most reliable metric by [71] – it captures both sensitivity and specificity and it doesn’t require arbitrary binarization of real-valued neuron activations. However, using correlation as the evaluation metric introduces two challenges that directly drive up total human study cost:

- **Challenge 1: High Labeling Cost.** Computing the correlation requires concept presence labels across the *entire* probing dataset. Labeling all inputs of a dataset with 50,000 images for a single (neuron, explanation) pair with three raters each costs roughly \$600, and evaluating a few thousand neurons in a single model can reach \$1M.
- **Challenge 2: Rater Noise.** Human concept judgments are noisy, and even small error rates can significantly affect correlation scores, particularly for rare concepts. Increasing

the number of raters per input improves reliability, but further multiplies overall cost.

In other words, *reliable, full-distribution human evaluation of neuron explanations is currently too expensive to perform at scale*. To address this, we introduce two novel techniques to enable cost-efficient and reliable crowd-sourced evaluation of automated interpretability.¹ In summary, our contributions are:

- **1. Model-Guided Importance Sampling (MG-IS) for Efficient Input Selection:** We develop an importance sampling strategy that selects the most informative inputs to show raters, reducing labeling cost by around 15× while preserving correlation estimation accuracy;
- **2. Bayes Rating Aggregation (BRAgg) for Robust Rater Modeling:** We propose a Bayesian rater uncertainty model that reduces the number of ratings required for stable correlation estimates by an additional 3×, further reducing the labeling cost;
- **3. Systematic Comparison of Automated Interpretability Methods:** Together, our proposed MG-IS and BRAgg reduce total evaluation cost by around 40×, taking the total costs of a systematic comparison down from around \$90,000 to \$2,160. Our work enables a large-scale, systematic comparison of leading neuron explanation methods across multiple vision models. Our findings demonstrate that Linear Explanations [69] overall produces the best vision neuron explanations, outperforming recent LLM-based explanations [4, 86].

4.2 Related Work

4.2.1 Automated Interpretability for Vision

Methods based on labeled concept datasets: Early work on automated interpretability sought to match neuron activations to human-interpretable concept annotations. Network

¹Our code is available at <https://github.com/Trustworthy-ML-Lab/Efficient-Interpretability-Eval>

Dissection (ND) [6] is the first automated method to generate text descriptions for individual neurons in vision models. They introduced the Broden dataset with dense pixel-wise annotations and proposed identifying concepts with high Intersection over Union (IoU) with binarized neuron activations. However, their approach requires dense concept annotated data which is expensive to collect. Compositional Explanations [65] extends Network Dissection to deal with polysemantic neurons, which may activate on multiple unrelated concepts, by searching for logical compositions of concepts such as "cat OR dog". Clustered Compositional Explanations (CCE)[54] extends this even further, addressing the problem that Compositional Explanations only explains the highest activations of a neuron. CCE divides the activation range of a neuron into 5 buckets, and generates a separate compositional explanation for each. [7] propose a version of Network Dissection utilizing a segmentation model instead of labeled data. INVERT[16] proposes explaining neurons with logical compositions of dataset class(or superclass) labels that maximize AUC, which removes the reliance on Broden but still requires labeled data.

Methods based on Language Models: Another popular line of work leverages generative language models to generate natural-language neuron descriptions. The first such method was MILAN [39], which trains a small generative neural network to describe the most highly activating inputs of a neuron. Describe-and-Dissect (DnD) [4] is a more recent method that utilizes pre-trained language models instead of training their own to generate detailed neuron descriptions based on highest activating inputs. Multimodal Automated Interpretability Agent (MAIA)[86] is another LLM based explanation pipeline where the explaining LLM agent can interact with many tools such as looking at highly activating inputs, generating new images or editing existing ones to generate its description.

CLIP-based methods: Finally, papers such as CLIP-Dissect (CD) [68] and FALCON [46] have proposed methods that don't require labeled concept information by relying on supervision from multimodal models such as CLIP [77]. Linear Explanations (LE) [69] proposes to explain polysemantic neurons as a linear combination of concepts, such as " $3 \times \text{dog} + 2 \times \text{cat}$ ". To obtain these explanations, they utilize either the class labels from the dataset denoted as LE(label), or

pseudo-labels from SigLIP [111] denoted as LE(SigLIP).

4.2.2 Human Eval of Automated Interpretability

Crowd-sourced human evaluations are commonly used to assess the quality of automatic neuron explanations in previous works [6, 68, 4]. These evaluations typically present raters the top-activating inputs for a neuron along with one or multiple candidate descriptions, and ask them to rate how well the description matches the top-activating inputs. However, as pointed out by NeuronEval [71], this evaluation protocol is flawed and fails to measure whether the description matches lower activations of the neuron, or whether all inputs corresponding to the description cause the neuron to fire, and in fact corresponds to only measuring *Recall*. As a result, this evaluation approach can favor explanations that are overly broad or non-specific.

On the other hand, in [12] and [117], the authors conducted more objective crowd-sourced evaluations of neuron interpretability by having raters predict whether an input is highly activating or not based on a feature visualization or a set of highly activating images for that neuron. However, these studies did not study text descriptions of neurons, and their methodology, which is roughly equivalent to measuring AUC, may also be biased towards overly generic neuron descriptions and does not pass the sanity checks proposed by [71].

NeuronEval [71] formalizes the task of Neuron evaluation under a unified mathematical framework which we utilize in this paper, and identifies flawed and good evaluation metrics through sanity tests. However, they do not conduct actual user studies using an evaluation metric that passes their sanity checks, which is the goal of this paper.

Many methods have been proposed to reduce noise in crowdsourced annotations, such as [79, 102], but they often require a consistent set of annotators (which we do not have) and do not leverage modern model capabilities.

4.3 Methods

In this section, we introduce two techniques to enable cost-efficient and reliable crowd-sourced evaluation of neuron explanations. Section 4.3.1 introduces Model-Guided Importance Sampling (MG-IS), which addresses the labeling cost challenge by selecting the most informative inputs to label. Section 4.3.2 introduces Bayes Rating Aggregation (BRAgg), which addresses the rater noise challenge by modeling rater errors to robustly aggregate binary judgments. Together, MG-IS and BRAgg substantially reduce the number of labels and ratings required for stable correlation estimates, making crowdsourced evaluation feasible in practice.

Problem Formulation. Given a neural network f and a neuron of interest k (or a direction in activation space), we can obtain a neuron explanation t (described by text) either manually or via automated interpretability methods, that aims to describe the role of that neuron. To measure how good this explanation is, we follow the framework introduced by [71], and use the Pearson correlation between the neuron activation vector a_k and the explanation concept presence vector c_t (see Fig. 4.1 for an overview of the evaluation pipeline). A high correlation coefficient means the concept is present on inputs where the neuron activates highly, and not present when neuron activates lowly.

Here, $a_k \in \mathbb{R}^{|\mathcal{D}|}$ is the activation vector of a neuron k over all the probing images $x_i \in \mathcal{D}$ and $c_t \in \mathbb{R}^{|\mathcal{D}|}$ is the concept vector to indicate if concept t appears on the probing images. The i -th component $[a_k]_i$ and $[c_t]_i$ can be formally expressed as: $[a_k]_i = f_k^{0:l}(x_i)$, $[c_t]_i = \mathbb{P}(t|x_i)$, $\forall i \in \{1, \dots, |\mathcal{D}|\}$ where $f^{0:l}$ denotes the neural network up to l -th layer. Our goal is to evaluate Pearson’s correlation coefficient ρ , which can be expressed as:

$$\rho(a_k, c_t) = \frac{1}{|\mathcal{D}|} \frac{\sum_{i \in \mathcal{D}} ([a_k]_i - \mu(a_k)) \cdot ([c_t]_i - \mu(c_t))}{\sigma(a_k)\sigma(c_t)} \quad (4.1)$$

where $\mu(a_k), \sigma(a_k), \mu(c_t), \sigma(c_t)$ are the mean and standard deviation of the vectors a_k and c_t .

4.3.1 Model-Guided Importance Sampling (MG-IS)

The activation vector a_k can be easily evaluated with a forward pass of the neural network. However, obtaining a high quality concept vector c_t can be expensive and difficult, requiring human raters or computationally expensive models. Our problem then becomes: *How can we best select a subset of inputs $S \subseteq \{|D|\}$ to label, such that we can accurately estimate $\rho(a_k, c_t)$ using just the subset?*

The simplest approach is to select input for annotation using uniform (Monte Carlo) sampling, where each input has an equal probability of being selected. However, concepts of interest could be rare in the probing dataset, meaning that a uniform sample is unlikely to contain sufficient amount (or any) of positive examples of the concept, making accurate estimation of correlation impossible.

Method: MG-IS

To address this issue, we propose estimating ρ using importance sampling, where we draw inputs x_i from a proposal distribution $\mathcal{Q}$ that places higher probability on informative inputs such as the ones that are likely to contain the concept, rather than from the original uniform distribution $\mathcal{P}$. Importantly, importance sampling preserves the expectation we seek to estimate, i.e. $\mathbb{E}_{x \sim \mathcal{P}}[h(x)] = \mathbb{E}_{x \sim \mathcal{Q}}[\frac{p(x)}{q(x)}h(x)]$ for any distribution $\mathcal{Q}$, given $q(x) > 0$ for all x such that $p(x)h(x) \neq 0$. Here $p(x)$ and $q(x)$ are the probability mass functions (pmf) of $\mathcal{P}$ and $\mathcal{Q}$ respectively.

According to [81] (Sec 3.3.2, Theorem 3.12, reproduced in Appendix D.3.5), when estimating the expected value of a function h , i.e. $\mathbb{E}_{x \sim \mathcal{P}}[h(x)]$, the optimal proposal distribution $\mathcal{Q}^*$ that minimizes the variance of the estimator should have the pmf $q^*(x) \propto |h(x)|p(x)$. Intuitively, this distribution samples more frequently from inputs that make a larger contribution to the value of the expectation, reducing the number of samples needed for a stable estimate.

To apply importance sampling to our correlation estimation, we first rewrite equation 4.1

in the form of an expectation over inputs:

$$\rho = \mathbb{E}_{x_i \sim \mathcal{D}} \frac{([a_k]_i - \mu(a_k)) \cdot ([c_t]_i - \mu(c_t))}{\sigma(a_k)\sigma(c_t)}, \quad (4.2)$$

Based on Theorem 1 and equation 4.2, we can estimate the ρ by sampling x_i from $\mathcal{Q}^*$ with pmf $q(x)$ as:

$$q^*(x_i) \propto \left| \frac{([a_k]_i - \mu(a_k)) \cdot ([c_t]_i - \mu(c_t))}{\sigma(a_k)\sigma(c_t)} \right| \cdot \frac{1}{|\mathcal{D}|} \quad (4.3)$$

$$\implies q^*(x_i) \propto |\bar{a}_{ki} \cdot \bar{c}_{ti}|, \quad (4.4)$$

where $\bar{a}_{ki}$ and $\bar{c}_{ti}$ are the normalized quantities $\frac{[a_k]_i - \mu(a_k)}{\sigma(a_k)}$ and $\frac{[c_t]_i - \mu(c_t)}{\sigma(c_t)}$ respectively. So to minimize the variance of estimator ρ , we should sample x_i more from the probing images that have high absolute product values $|\bar{a}_{ki} \cdot \bar{c}_{ti}|$.

Since we do not know c_t before running the test (i.e. through human study or computationally expensive model), we instead use a cheaper (and less accurate) method to approximate c_t for the purposes of sampling. In particular, we use the SigLIP [111] model to predict c_t , which we denote as c_t^{siglip} . This gives us the q^{siglip} sampling distribution:

$$q^{siglip}(x_i) = \frac{|[\bar{a}_k]_i \cdot [\bar{c}_t^{siglip}]_i|}{\sum_{x_i \in \mathcal{D}} |[\bar{a}_k]_i \cdot [\bar{c}_t^{siglip}]_i|} \quad (4.5)$$

To account for the error in siglip estimates, and to ensure that importance sampling is unbiased ($q(x) > 0$ when $p(x)h(x) \neq 0$), our final sampling distribution $q^{\text{MG-IS}}(x)$ is a mixture of $q^{siglip}(x)$ and the original distribution $p(x) = \frac{1}{|\mathcal{D}|} \forall x \in \mathcal{D}$:

$$q^{\text{MG-IS}}(x) = (1 - \gamma)q^{siglip}(x) + \gamma p(x) \quad (4.6)$$

The hyperparameter γ was chosen experimentally, with best performance using $\gamma = 0.2$. See Appendix D.2.1 for details. This is equivalent to selecting 80% of the samples from q^{siglip} and 20% of them from the uniform distribution. We name this approach model-guided importance

sampling (MG-IS) because the sampling is guided by the SigLIP models.

Sampling Correction. To obtain an unbiased estimate of the correlation using importance sampling, we must estimate not only the final coefficient, but also the mean and variance of the concept vector c_t . To evaluate them we apply the importance sampling correction $\frac{p(x)}{q(x)}$ sequentially. We first estimate the mean $\mu_S(c_t)$, and then the standard deviation $\sigma_S(c_t)$ from the samples $x_i \sim \mathcal{Q}$:

$$\mu_S(c_t) = \frac{1}{|S|} \sum_{i \in S} \frac{p(x_i)}{q(x_i)} [c_t]_i \quad (4.7)$$

$$\sigma_S(c_t) = \sqrt{\frac{1}{|S| - 1} \sum_{i \in S} \frac{p(x_i)}{q(x_i)} ([c_t]_i - \mu_S(c_t))^2}. \quad (4.8)$$

Next, we normalize the inputs: $[\bar{c}_t]_S = \frac{[c_t]_S - \mu_S(c_t)}{\sigma_S(c_t)}$. For a_k , since we have access to the entire neuron activation vector, we can directly compute the true mean and standard deviation (i.e. $\mu(a_k)$, $\sigma(a_k)$) instead of needing to estimate them from the sample. Thus, we have $[\bar{a}_k]_S = \frac{[a_k]_S - \mu(a_k)}{\sigma(a_k)}$. Finally, we obtain the estimated correlation score of ρ_S as:

$$\rho_S = \frac{1}{|S|} \sum_{i \in S} \frac{p(x_i)}{q(x_i)} [\bar{a}_k]_i \cdot [\bar{c}_t]_i, \quad (4.9)$$

where we use the subscript S to denote that the correlation score is estimated on the subset S .

Simulation study: Sampling Effectiveness

To study the accuracy of our estimated correlation ρ_S , we simulate a human study using existing labeled data. We then simulate limited data by only using a subset of these labels, and compare different sampling methods based on how close their estimated correlation coefficient ρ_S is to the true correlation coefficient ρ_{gt} in equation 4.11.

In particular, we use the ImageNet[19] dataset and its class and superclass labels, which we denote as the ground truth concept labels c_t^* . We first generate descriptions t_k for all 2048 neurons in layer4 of ResNet-50[38] trained on ImageNet, by selecting the concept t_k from ImageNet class and superclass names $\mathcal{C}$ that maximizes correlation with neuron k 's activations.

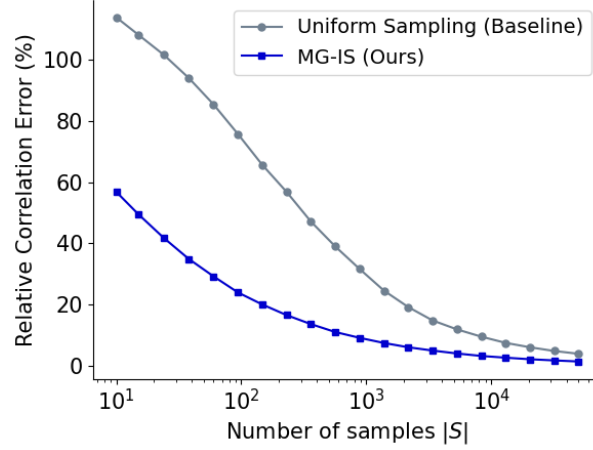


Figure 4.2. Comparing different sampling strategies to estimate ρ_S . Our Model-Guided Importance Sampling (MG-IS) using SigLIP estimates significantly outperform baseline regardless of the size of sampled subset.

$$t_k = \operatorname{argmax}_{t \in \mathcal{C}} \rho(a_k, c_t^*) \quad (4.10)$$

$$\rho_{gt} = \rho(a_k, c_{t_k}^*) \quad (4.11)$$

To compare different sampling strategies, we sample subsets of observations using their proposed distributions Q , and measure the Relative Correlation Error (RCE, Eq. 4.12) compared to correlation calculated from full distribution:

$$RCE(Q, |S|) = \frac{\sum_{k \in K} \mathbb{E}_{S \sim Q} [|\rho_S(a_k, c_{t_k}) - \rho_{gt}(a_k, c_{t_k}^*)|]}{\sum_{k \in K} |\rho_{gt}(a_k, c_{t_k}^*)|} \quad (4.12)$$

For Uniform Sampling, $Q = P$. In practice we evaluate the expectation by averaging over 10 samples from Q .

Results. Figure 4.2 shows the Relative Correlation Error of different strategies as a function of number of samples $|S|$. We can see that overall our MG-IS, as defined in equation 4.6, performs much better – which has low error and lower cost. Compared to the Uniform Sampling baseline, we can match the correlation estimation accuracy with around $15\times$ fewer samples. Alternatively, we can reach $\sim 65\%$ lower correlation estimation error with the same budget.

In Appendix D.2.3 we also experiment with a version of importance sampling without model guidance, based on a_k only, and show it can still be useful (better than uniform sampling) for the cases where no cheap model is available, but overall it performs worse than the MG-IS.

4.3.2 Bayes Rating Aggregation (BRAgg)

Another challenge in human evaluation is that the concept labels c_t obtained are noisy due to variability in rater attention, interpretation, and experience – this is particularly a problem when using crowd-sourced platforms such as Amazon Mechanical Turk. Such noise can significantly affect correlation estimation, particularly when the concept is rare as we might end up with more false positives than true positives.

To mitigate this, for each input x_i and concept t , we collect m independent binary ratings $r_{ti}^j \in \{0, 1\}, j \in \{1, \dots, m\}$ from different raters. Let $R_{ti} = \{r_{ti}^1, \dots, r_{ti}^m\}$ denote the set of ratings for a particular (input, concept) pair (x_i, t) . The goal of rating aggregation is to combine these m judgments for each input x_i into a single estimated concept presence value $[c_t]_i$. Below, we describe three different methods for Rating Aggregation to get $[c_t]_i$.

Method 1 - Average: $[c_t]_i = \frac{\sum_{j=1}^m r_{ti}^j}{m}$

$\Rightarrow$ Ratings are simply averaged to form c_t .

Method 2 - Majority Vote:

$$[c_t]_i = \begin{cases} 1, & \text{if } \frac{\sum_{j=1}^m r_{ti}^j}{m} > 0.5; \\ 0, & \text{if } \frac{\sum_{j=1}^m r_{ti}^j}{m} \leq 0.5. \end{cases}$$

$\Rightarrow$ The most common rating becomes the label $[c_t]_i$.

While these methods are simple and commonly used, they may not be the optimal ways to model our confidence that a concept is present in a particular input. To better capture this uncertainty, we propose Bayes Rating Aggregation (BRAgg) to estimate $[c_t]_i$ as the posterior

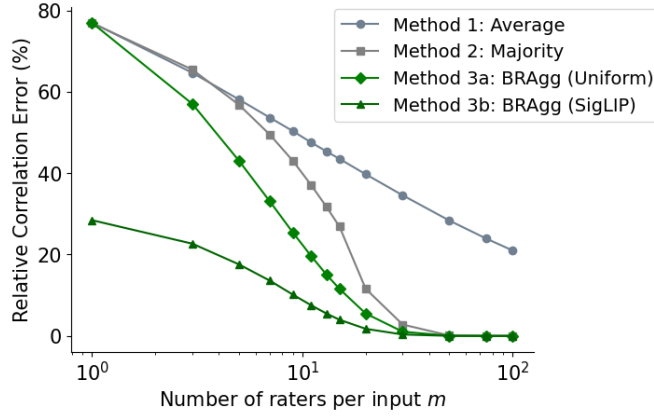


Figure 4.3. Comparison of different rating aggregation strategies on simulated human study with error rate $\eta = 23\%$, tested on the full dataset.

probability that the concept is present: $[c_t]_i = \mathbb{P}([c_t^*]_i = 1 | R_{ti})$. Here c_{ti}^* denotes the "ideal" or true concept value without any labeling noise.

Method 3 - Bayes Rating Aggregation (BRAg):

Let $C_{ti} := ([c_t^*]_i = 1)$ and $\neg C_{ti} = ([c_t^*]_i = 0)$, then:

$$[c_t]_i = \frac{\mathbb{P}(R_{ti} | C_{ti}) \cdot \mathbb{P}(C_{ti})}{\mathbb{P}(R_{ti} | C_{ti}) \cdot \mathbb{P}(C_{ti}) + \mathbb{P}(R_{ti} | \neg C_{ti}) \mathbb{P}(\neg C_{ti})} \quad (4.13)$$

where we first use Bayes rule to expand the posterior $[c_t]_i = \mathbb{P}(c_{ti}^* = 1 | R_{ti})$ to get equation 4.13, and then calculate the Likelihood and Prior terms in equation 4.13 as follows:

(I) Likelihood: $\mathbb{P}(R_{ti} | C_{ti})$. We assume each rater makes errors uniformly at random with error rate η , i.e. for any input $\mathbb{P}(r_{ti}^j = c_{ti}^*) = 1 - \eta$, where η is a parameter we can estimate experimentally. Let $\alpha_{ti} = \sum_{j=1}^m r_{ti}^j$, we obtain the likelihood in below equations:

$$\mathbb{P}(R_{ti} | C_{ti}) = (1 - \eta)^{\alpha_{ti}} (\eta)^{(m - \alpha_{ti})} \quad (4.14)$$

$$\mathbb{P}(R_{ti} | \neg C_{ti}) = (\eta)^{\alpha_{ti}} (1 - \eta)^{(m - \alpha_{ti})} \quad (4.15)$$

Note we also tested more advanced modeling where different (input, concept) pairs can have different error rates to account for some inputs being more ambiguous/harder to label, but we

found that simple uniform error modeling performed slightly better overall. See Appendix D.2.4 for full results.

(II) Prior: $\mathbb{P}(C_{ti})$. There are multiple ways of choosing priors, reflecting the confidence of whether a concept exists in an input x_i . In our analysis, we consider two different priors:

- **(a) Uniform Prior:** for all concepts t and all inputs x_i , set $\mathbb{P}(C_{ti}) = \beta$
- **(b) SigLIP Prior:** we leverage knowledge from a cheap evaluator, namely SigLIP to initialize the prior and set $\mathbb{P}(C_{ti}) = [c_t^{siglip}]_i$

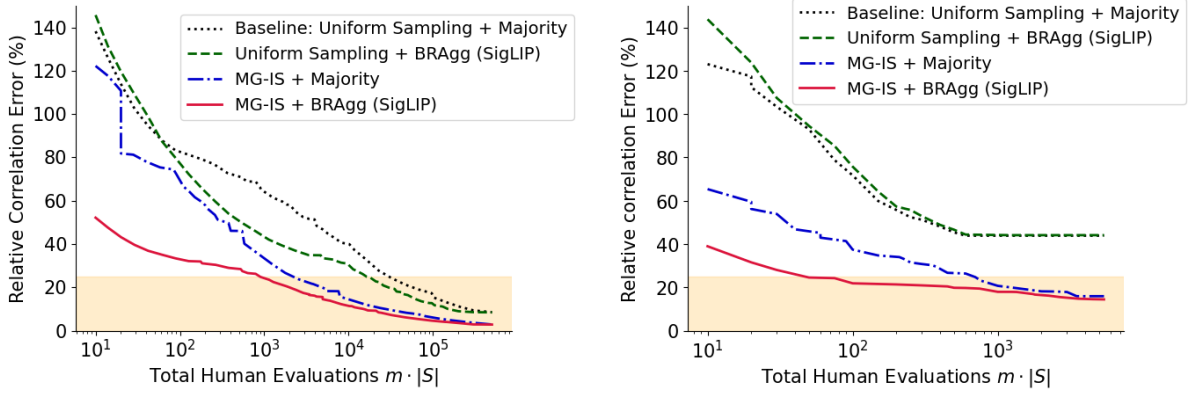
Here β is a hyperparameter, which is set as 0.05. We evaluate the effects of different choices for β in Appendix D.2.2. For the SigLIP prior, we clip the prior to be between 0.001 and 0.999 to avoid extreme values which may dominate the final result. Our method with SigLIP prior can be seen as a hybrid evaluation that combines both human and model knowledge.

Simulation: Effect of Rating Aggregation Method

Setup:

We follow the ImageNet based setup described in Section 4.3.1 for defining the ground truth correlation ρ_{gt} (Eq. 4.11). However, instead of using a subset of inputs, in this test we evaluate how accurately we can estimate the correlation with access to m noisy ratings for each input. For our testing, we used an error rate $\eta = 23\%$, i.e. randomly flipping 23% of the labels, which aligns with our experimental observations. We report the Relative Correlation Error (RCE) between estimated correlations and ρ_{gt} in Figure 4.3.

As we can see from Figure 4.3, Method 3b BRagg(SigLIP) resulted in clearly the lowest RCE, allowing us to reach relatively accurate estimates with just a few raters, while other methods require > 10 raters per input before reaching low error rates. Overall BRagg (Uniform) performs the second best, while Average rating aggregation fails to reach low errors even with 100 raters per input. Depending on the desired error rate, our BRagg(SigLIP) can achieve it with 2-10 $\times$ fewer raters than the best baseline (majority).



(a) Setting 1: Simulation with 23% label noise.

(b) Setting 2: MTurk method validation.

Figure 4.4. Comparing the effect of our sampling technique MG-IS, rating aggregation method BRagg independently and both together. The shaded region represents $RCE \leq 25\%$.

4.4 Experimental Validation and Human Study Design: Combining MG-IS and BRagg

To further understand the performance of our proposed techniques, and to determine optimal parameters for our large scale human study in Section 4.5, we tested our methods in two realistic settings with both label noise and limited samples.

Setting 1 - Simulation: Our first setting is based on the simulation setups described in Section 4.3, but this time with *both* reduced samples and 23% label noise. The results of this setting are shown in Figure 4.4a.

Setting 2 - Crowd-sourced Validation Study: In this setting, we conducted a *real* crowd-sourced evaluation on AMT to annotate the concept labels c_t for samples x_i in the probing dataset, on inputs where we also have access to the ground-truth labels c_t^* from the dataset for comparison. Specifically, we performed a crowdsourced evaluation of 10 neurons in Resnet-50 layer4 with explanations selected from ImageNet classes and superclasses following Eq. equation 4.10. We then estimated the correlation coefficient ρ_S using human ratings R on the rated subset S and measured how close our estimate is to the ground truth ρ_{gt} . This setup allows us to evaluate the error rate on Mechanical Turk and the effectiveness of our techniques on real crowdsourced data.

We randomly selected 10 neurons (excluding neurons with very vague concepts like *artifact*) and labeled 600 inputs per neuron with 9 AMT raters labeling each input. To estimate the evaluation accuracy with smaller number of raters/inputs we randomly sampled subsets of the ratings for our plots in Figure 4.4b (Note: each point is the average of 1000 random samples of subsets of the same size). We also used this setting to estimate the error rate of MTurk raters as $\eta = \mathbb{P}(r_{ti}^j \neq [c_i^*]_i)$, giving us an estimate of 23%, which we used for our simulations. We note this might slightly overestimate the noisiness of raters as some ImageNet labels are in reality ambiguous or even incorrect, but it serves as a useful approximation.

Results: Figure 4.4 shows the results of different sampling and rating aggregation techniques across our two settings. The x-axis of Figure 4.4 shows the total number of ratings i.e. number of samples $|S|$ times number of ratings per input m . Since the optimal number of ratings per input depends on the total budget (for smaller budgets it is better to use fewer raters per sample), we plot the Pareto-front i.e. the optimal number of raters for each budget in Figure 4.4.

From the results we can see that by far the best results are achieved when using both MG-IS and BRagg together (red line), while they also help individually with importance sampling being more important overall. In our Simulation setting (Fig. 4.4a), with a budget of 550 evaluations per neuron, we can reach RCE of 27.5% with MG-IS + BRagg. To reach the same error rate requires 1760 ($3\times$) evaluations with MG-IS only, 14607 ($25\times$) samples with BRagg only and 22560 ($40\times$) samples using the baseline uniform sampling with majority vote, showing our techniques lead to around $40\times$ cost reduction overall.

On the MTurk evaluation (Fig. 4.4b), overall error rates are lower, but the trends between methods are similar, with MG-IS + BRagg reaching 19.8% RCE within the budget of 550 evaluations per neuron, while MG-IS only requires 2000 evaluations to reach the same error ($4\times$). Uniform sampling methods never reach sufficiently low errors within the number of samples we evaluated, but extrapolating from current results we estimate it would require 10k-30k evaluations to reach 20% RCE.

Based on our findings and our budget, we choose to use 3 raters per input and to evaluate

180 inputs per neuron for 540 evaluations per neuron in the large scale study in Section 4.5. As we can evaluate 15 inputs in one task for the cost of \$0.06, this evaluation will cost us $\frac{\$0.06}{15} \cdot 180 \cdot 3 = \2.16 per (neuron, explanation) pair. See Appendix D.1.3 for detailed results on the tradeoff between more inputs vs. more ratings per input, as well as how to use these experiments to decide the optimal parameters for our large-scale study.

Accuracy of SigLIP-only evaluation. Since our techniques are assisted by the SigLIP model, it is natural to ask how does it compare against fully automated evaluation? Our techniques can be seen as a hybrid method combining the strengths of human ratings and efficiency of model-based evaluation. Overall, our results indicate SigLIP-based evaluation is quite reliable by itself. When measuring the RCE of SigLIP only (using the full dataset), it reaches RCE of 32.10% on all neurons (vs. our estimated human study RCE of 27.90%), and on the subset of 10 neurons used in our MTurk validation SigLIP only RCE is 22.47% compared to 19.8% of our human study. So overall SigLIP only evaluation has around 15% higher error than our human study. This is still more accurate than a low budget human-study, and as such can be a good practical choice for future studies. However, it is important to remember that human evaluation is the gold standard for interpretability, and we should not blindly rely on automated evaluation without validating them with experiments such as ours. See Appendix D.2.5 for a more detailed analysis of SigLIP evaluation accuracy.

4.5 Large-Scale Crowdsourced Study

Setup. Our study focuses on neurons in two layers of two different models: Layer4 of ResNet-50 trained on ImageNet, and MLP activations in Layer11 of ViT-B-16 [20] trained on ImageNet. We use the full ImageNet Validation set as probing dataset, see Appendix D.3.3 for additional details.

Table 4.1. SigLIP based simulation with correlation scoring comparing different simple explanation methods ($l = 1$). We can see Linear Explanation (LE) performs the best even when restricted to single concept explanations.

Simple Exp. ($l=1$)	ND [7]	MILAN [39]	CD [68]	INVERT $l=1$ [16]	DnD [4]	MAIA [86]	LE(label) $l=1$ [69]	LE(SigLIP) $l=1$ [69]
RN-50 (Layer4)	0.1242 ± 0.002	0.0920 ± 0.002	<u>0.1904</u> <u>± 0.002</u>	0.1867 ± 0.002	0.1534 ± 0.002	0.1396 ± 0.009	0.1793 ± 0.002	0.2413 ± 0.002
ViT-B-16 (Layer11 MLP)	0.0335 ± 0.002	0.0194 ± 0.002	0.1849 ± 0.004	0.1343 ± 0.004	0.1049 ± 0.003	0.1497 ± 0.021	<u>0.2704</u> <u>± 0.004</u>	0.2968 ± 0.004

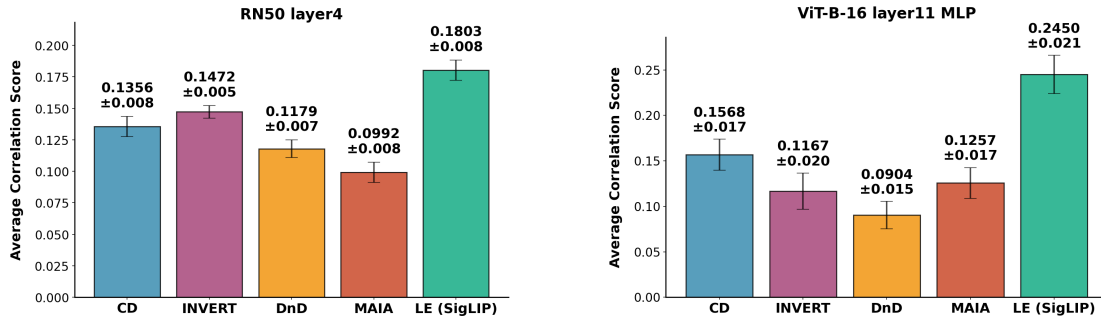


Figure 4.5. Results of our MTurk study. Using BRagg(SigLIP) for rating aggregation. All methods are restricted to length 1 explanations. Overall explanations of LE(SigLIP) have the highest correlation coefficient.

4.5.1 Method Selection via Automated Evaluation

Many different methods have been introduced to explain neurons in vision models. Due to limited budget, we are unable to compare all these methods on our crowdsourced study, so we first perform an automated evaluation to find the best methods to include in the crowdsourced study.

Explanation Complexity: Some methods explain a single neuron as a combination of concepts. For example [65, 16] use a logical composition of explanations, such as "dog OR cat". For fairness and simplicity, in our main experiments we focus on explanations with only one concept $l = 1$, but we also conduct an automated evaluation of complex explanations in Appendix D.2.6.

For automated evaluation we use the SigLIP based simulation with correlation scoring introduced by [69], see Appendix D.3.4 for details. The results of this automated evaluation are

shown in Table 4.1.

Overall we can see Linear Explanations [69](even when restricted to a single concept) produces the highest correlation scores. Other well performing methods include INVERT [16] and CLIP-Dissect [68], followed by recent language model based explanations MAIA [86] and DnD[4]. We observed that Broden based methods [6, 65, 54] overall performed relatively poorly, likely because their concept sets do not include the relevant higher level concepts such as animal species needed to describe later neurons of ImageNet models.

4.5.2 Crowdsourced Evaluation

Details: Based on automated evaluation results, we chose to conduct our crowdsourced study on the 5 best performing simple explanations: LE (SigLIP)[69], CLIP-Dissect[68], INVERT[16], MAIA[86] and DnD[4]. We evaluated 100 randomly selected neurons for each of the two models, and 180 inputs per neuron with 3 raters each. This gives us a total of 1000 (neuron, explanation) pairs to evaluate for a total cost of \$2160. Our study was deemed Exempt from IRB approval by the IRB review board at our institution.

User Interface. The main objective of our crowdsourced study is to have users annotate which inputs have concept t present. We do this by presenting raters with explanation t and 15 inputs per task, and asking them to select inputs where concept is present. The full study interface and additional details are available in Appendix D.3.2.

Results

The overall results from the crowdsourced evaluation are shown in Figure 4.5. See Appendix D.3.2 for more detailed results, including example explanations and their scores. We can see Linear Explanations performs the best overall despite being limited to using a predefined concept set. We think this is likely because it is the only explanation method tested that is optimized to explain the entire range of activations instead of focusing only on highest activations. The second best performing method is CLIP-Dissect [68] or INVERT [16] depending

on the model evaluated. It is somewhat surprising that recent methods based on large language models [4, 86] did not outperform simpler baselines despite being able to provide very accurate and complex descriptions in some cases. We think this is caused by a few reasons:

1. Focus on highly activating inputs only, leading to overly specific explanations that do not describe lower activations.
2. Inconsistency. While all 5 methods mostly produce relevant descriptions, more complex methods such as those relying on LLMs often have more variance in their description quality, leading to poor explanations for some neurons.

Overall the correlation scores were quite low, with best methods reaching ~ 0.2 correlation, highlighting the need for further improvement in explanation methods and/or creating more interpretable models.

4.6 Conclusion

In this paper, we introduced two new techniques for conducting principled crowdsourced evaluation of neuron explanations. First, we proposed Model-Guided Importance Sampling (MG-IS) to determine which inputs to show raters, reducing our labeling cost by $\sim 15\times$. Second, we introduced a new Bayes-rule based method (BRAGg) to aggregate multiple ratings, which further reduces the amount of ratings needed to reach a certain level of accuracy by $\sim 3\times$. Finally, using these methods, we were able to conduct a large-scale human study with reasonable cost (lowering cost from \$90k to \$2.16k USD) without sacrificing accuracy – We evaluated 5 SOTA automated interpretability methods and discovered that Linear Explanations [69] performed the best overall.

Acknowledgement

The authors are partially supported by National Science Foundation under Grant No. 2107189, 2313105, 2430539, Hellman Fellowship, Intel Rising Star Faculty Award. The authors would like to thank anonymous reviewers for valuable feedback to improve the manuscript.

Chapter 4, in full, is based on the paper from Tuomas Oikarinen, Ge Yan, Akshay Kulkarni and Tsui-Wei Weng, "Beyond Top Activations: Efficient and Reliable Crowdsourced Evaluation of Automated Interpretability" published at the Conference on Computer Vision and Pattern Recognition (CVPR), 2026. The dissertation author is the primary investigator and author of this paper.

Chapter 5

Label-Free Concept Bottleneck Models

5.1 Introduction

Deep neural networks (DNNs) have demonstrated unprecedented success in a wide range of machine learning tasks such as computer vision, natural language processing, and speech recognition. However, due to their complex and deep structures, they are often regarded as *black-box* models that are difficult to understand and interpret. Interpretable models are important for many reasons such as creating calibrated trust in models, which means understanding when we should trust the models. Making deep learning models more interpretable is an active yet challenging research topic.

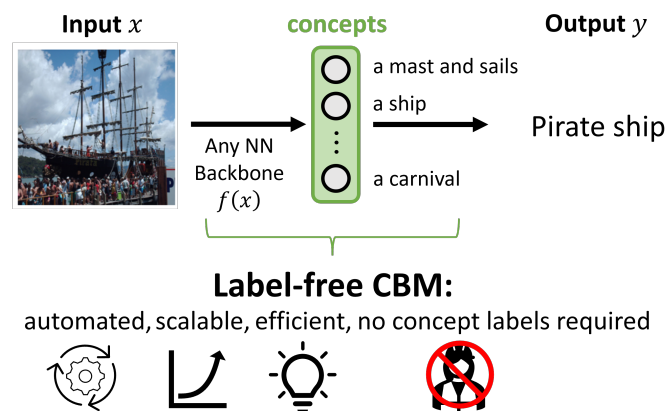


Figure 5.1. Our proposed Label-free CBM has many desired features which existing CBMs lack, and it can transform any neural network backbone into an interpretable Concept Bottleneck Model.

One approach to make deep learning more interpretable is through Concept Bottleneck Models (CBMs) [50]. CBMs typically have a Concept Bottleneck Layer before the (last) fully connected layer of the neural network. The concept bottleneck layer is trained to have each neuron correspond to a single human understandable *concept*. This makes the final decision a linear function of interpretable concepts, greatly increasing our understanding of the decision making. Importantly, CBMs have been shown to be useful in a variety of applications, including model debugging and human intervention on decisions. However, there are two crucial limitations of existing CBMs and their variants [50, 108, 116]: (i) labeled data is required for each of the predefined concepts, which is time consuming and expensive to collect; (ii) the accuracy of a CBM is often significantly lower than the accuracy of a standard neural network, especially on more complex datasets.

To address the above two challenges, we propose a new framework named Label-free CBM, which is capable of transforming any neural network into an interpretable CBM *without* labeled concept data while *preserving* accuracy comparable to the original neural network by leveraging foundation models [11]. Our Label-free CBM has many advantages:

- *Scalable* - to our best knowledge, it is the first CBM that scales to ImageNet
- *Efficient* - creating a CBM takes only a few hours even for very large datasets
- *Automated* - training it for a new dataset requires minimal human effort

5.2 Related Work

Post-hoc explanations [84]: The approach of post-hoc explanations includes some classic methods such as LIME [80] and SHAP [58], which try to explain individual model decisions by identifying which parts of the input data (e.g. pixels) are the most important for a given decision. However, these methods are based on local approximations of the DNN model and as such are not always accurate. Further, the explanations at the granularity of input pixels may not always be helpful and could require substantial subjective analysis from humans. In

contrast, our explanations in Section 5.4 are not approximated and explain predictions in terms of human-understandable concepts.

More interpretable final layer: [103] proposes making the FC layer sparse, and develop an efficient algorithm for doing so. They show that sparse models are more interpretable in many ways, but it still suffers from the fact the previous layer features are not interpretable. NBDT [99] proposes replacing the final layer with a neural backed decision tree for another form of more interpretable decisions. Other approaches to make neural networks more interpretable include Concept Whitening [17] and Concept Embedding Models [109].

CBM: Most related to our approach are Concept Bottleneck Models [50, 57] which create a layer before the last fully connected layer where each neuron corresponds to a human interpretable concept. CBMs have been shown to be beneficial by allowing for human test-time intervention for improved accuracy, as well as being easier to debug. To reduce the training cost of a CBM, a recent work [108] proposed Post-Hoc CBM that only needs to train the last FC layer along with an optional residual fitting layer, avoiding the need to train the backbone from scratch. This is done by leveraging Concept Activation Vectors (CAV) [48] or the multi-modal CLIP model [77]. However, the post-hoc CBM does not fully address the problems of the original CBM as using TCAV still requires collecting annotated concept data and their use of CLIP model can only be applied if the NN backbone is the CLIP image encoder. Additionally, the performance of post-hoc CBMs without uninterpretable residual fitting layers is often significantly lower than the standard DNNs. Similarly, an earlier work Interpretable Basis Decomposition [116] proposes learning a concept bottleneck layer based on labeled concept data for explainable decisions, even though they do not call themselves a CBM. Comparison between the features of our method and existing approaches is shown in Table 5.1.

Model editing/debugging: Our approach is related to a range of works proposing ways to edit networks, such as [5, 100] for generative vision models, [5] for classifiers, or [60, 62] for language models. In addition [1] propose a way to debug model mistakes using TCAV activation vectors.

Table 5.1. Comparison of our method against existing methods for creating Concept Bottleneck models, CBM [50], IBD [116] and 4 versions of P-CBM [108], where ‘-h’ indicates the hybrid model that uses uninterpretable residual term, ‘(CLIP)’ means models using CLIP concepts. We used “Maybe” to indicate models that could in theory extend to ImageNet but have not been tested.

Method:	(I) Flexibility		(II) Interpretability		(III) Performance	
	Without labeled concept data	Any network architecture	Sparse final layer	All features interpretable	Preserves accuracy	Extends to ImageNet scale
CBM	No	Yes	No	Yes	No	No
IBD	No	Yes	No	No	Yes	No
P-CBM	No	Yes	Yes	Yes	No	No
P-CBM (CLIP)	Yes	No	Yes	Yes	No	Maybe
P-CBM-h	No	Yes	Yes	No	Yes	No
P-CBM-h (CLIP)	Yes	No	Yes	No	Yes	Maybe
Label-free CBM (This work)	Yes	Yes	Yes	Yes	Yes	Yes

CLIP-Dissect [68]: CLIP-Dissect is a recent method for understanding the roles of hidden layer neurons by leveraging the CLIP multimodal model [77]. It can provide a score of how close any neuron is to representing any given concept without the need of concept annotation data, which makes it useful as an optimization target for learning our interpretable projection in Step 3.

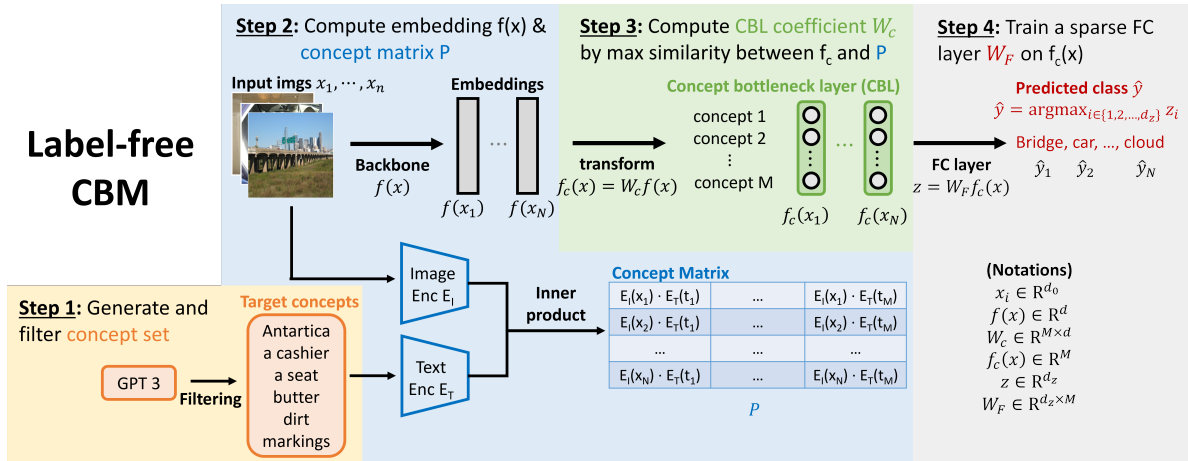


Figure 5.2. Overview of our pipeline for creating Label-free CBM.

5.3 Label-free CBM: A new framework to build CBM

In this section, we propose Label-free CBM, a novel framework that builds a concept bottleneck model (CBM) in an *automated*, *scalable* and *efficient* fashion and addresses the core limitations of existing CBMs. Given a neural network backbone, Label-free CBM transforms the backbone into an interpretable CBM without the need of concept labels with the following 4 steps which are illustrated in Fig 5.2 – **Step 1:** Create the initial concept set and filter undesired concepts; **Step 2:** Compute embeddings from the backbone and the concept matrix on the training dataset; **Step 3:** Learn projection weights W_c to create a Concept Bottleneck Layer (CBL); **Step 4:** Learn the weights W_F of the sparse final layer to make predictions.

Note that the backbone model can either be a model trained on the target task, or a general model trained on a different task. The details of Label-free CBM for Step 1 are provided in Sec. 5.3.1, Step 2 and Step 3 in Sec. 5.3.2, and finally Step 4 in Sec. 5.3.3.

5.3.1 Step 1: Concept set creation and filtering

In this step, we will describe how to create a concept set to serve as the basis of human-interpretable concepts in the Concept Bottleneck Layer. This step consists of two sub-steps: **A. Initial concept set creation** and **B. Concept set filtering**.

A. Initial concept set creation: A concept set refers to the set of concepts represented in the Concept Bottleneck Layer. In the original CBM paper [50], this is decided by domain experts as the set of concepts that are important for the given task. However, since our objective is to automate the entire process of generating CBMs, we don't want to rely on human experts. Instead, we propose generating the concept set via GPT-3 [15] using the OpenAI API. Somewhat surprisingly, GPT-3 has a good amount of domain knowledge of which concepts are important for detecting each class when prompted in the right way. Specifically, we ask GPT-3 the following:

- List the most important features for recognizing something as a {class}:
- List the things most commonly seen around a {class}:

- Give superclasses for the word {class}:

Note that the {class} in here refers to the class name in the machine learning task. For GPT-3 to perform well on the above prompt, we provide two examples of the desired outputs for few-shot adaptation. Note that those two examples can be shared across all datasets, so no additional user input is needed for generating a concept set for a new dataset. Full prompts and example outputs are illustrated in the Appendix Figures E.4 and E.5. To reduce variance we run each prompt twice and combine the results. Combining the concepts received from different classes and prompts gives us a large, somewhat noisy set of initial concepts, which we further improve by filtering. We found using GPT-3 to generate initial concepts to perform better than using the knowledge-graph ConceptNet [90] which was used in Post Hoc-CBM [108]. Reasons for this and a comparison between the two are presented in Appendix E.1.6.

B. Concept set filtering: Next we employ several filters to improve the quality and reduce the size of our concept set, as stated below:

1. *Concept length:* We delete any concept longer than 30 characters in length, to keep concepts simple and avoid unnecessary complication.
2. *Remove concepts too similar to classes:* We don't want our CBM to contain the output classes themselves as that would defeat the purpose of an explanation. To avoid this we remove all concepts that are too similar to the names of target classes. We measure this with cosine similarity in a text embedding space. In particular we use an ensemble of similarities in the CLIP ViT-B/16 text encoder as well as the all-mpnet-base-v2 sentence encoder space, so our measure can be seen as a combination of visual and textual similarity. For all datasets, we deleted concepts with similarity > 0.85 to any target class.
3. *Remove concepts too similar to each other:* We also don't want duplicate or synonymous concepts in the bottleneck layer. We use the same embedding space as above, and remove any concept that has another concept with > 0.9 cosine similarity already in concept set.

4. *Remove concepts not present in training data:* To make sure our concept layer accurately presents its target concepts, we remove any concepts that don't activate CLIP highly. This cut-off is dataset specific, and we delete all concepts with average top-5 activation below the cut-off.
5. *Remove concepts we can't project accurately:* Remove neurons that are not interpretable from the CBL. This step is actually performed after step 3 and is described in Section 5.3.2.

We discuss the reasoning behind the filters in detail in Appendix E.1.5 and perform an ablation study on their effects in Appendix E.1.4.

5.3.2 Steps 2 and 3: Learning the Concept Bottleneck Layer (CBL)

Once the concept set is obtained, the next step is to learn a projection from the backbone model's feature space into a space where axis directions correspond to interpretable concepts. Here, we present a way of learning the projection weights W_c without any labeled concept data by utilizing CLIP-Dissect [68]. To start with, we need a set of target concepts that the bottleneck layer is expected to represent as $\mathcal{C} = \{t_1, \dots, t_M\}$, as well as a training dataset (e.g. images) $\mathcal{D} = \{x_1, \dots, x_N\}$ of the original task. Next we calculate and save the CLIP concept activation matrix P where $P_{i,j} = E_I(x_i) \cdot E_T(t_j)$ and E_I and E_T are the CLIP image and text encoders respectively. W_c is initialized as a random $M \times d_0$ matrix where d_0 is the dimensionality of backbone features $f(x)$. The initial set $\mathcal{C}$ is created in Step 1 and the training set $\mathcal{D}$ is provided by the downstream task. We define $f_c(x) = W_c f(x)$, where $f_c(x_i) \in \mathbb{R}^M$. We use k to denote a neuron of interest in the projection layer, and its activation pattern is denoted as q_k where $q_k = [f_{c,k}(x_1), \dots, f_{c,k}(x_N)]^\top$, with $q_k \in \mathbb{R}^N$ and $f_{c,k}(x) = [f_c(x)]_k$.

To make the neurons in the CBL interpretable, we need to enforce the projected neurons to activate in correlation with the target concept, which we do by optimizing W_c to maximize the CLIP-Dissect similarity between the neuron's activation pattern and the target concept. To

optimize this similarity, we have designed a new fully differentiable similarity function $\text{sim}(t_i, q_i)$ that can be applied to CLIP-Dissect, called *cos cubed*, that still achieves very good performance in explaining the neuron functionality as shown in Appendix E.1.3. Our optimization goal is to minimize the objective L over W_c as defined in Equation (5.1):

$$L(W_c) = \sum_{i=1}^M -\text{sim}(t_i, q_i) := \sum_{i=1}^M -\frac{\bar{q}_i^3 \cdot \bar{P}_{:,i}^3}{\|\bar{q}_i^3\|_2 \|\bar{P}_{:,i}^3\|_2}. \quad (5.1)$$

Here $\bar{q}$ indicates vector q normalized to have mean 0 and standard deviation 1, and the *cos cubed* similarity $\text{sim}(t_i, q_i)$ is simply the cosine similarity between two activation vectors after both have been normalized and raised to third power element-wise. The third power is necessary to make the similarity more sensitive to highly activating inputs. As this is still a cosine similarity, it takes values between $[-1, 1]$. We optimize $L(W_c)$ using the Adam optimizer on training data $\mathcal{D}$, with early stopping when similarity on validation data starts to decrease. Finally to make sure our concepts are truthful, we drop all concepts j with $\text{sim}(t_j, q_j) < 0.45$ on validation data after training W_c . This is the 5th concept set filter from Sec 5.3.1. This cutoff was selected manually as a good indicator of a neuron being interpretable. During this filtering, the number of concepts is reduced: $M \leftarrow M - \Delta$, where Δ is non-negative integer representing the number of concepts being removed in this step. Note that matrix W_c has also to be updated accordingly by removing the rows that correspond to the removed concepts, and with our notation, $W_c \in \mathbb{R}^{M \times d_0}$. To simplify Figure 5.2, we omit plotting this concept removal step.

5.3.3 Step 4: Learning the sparse final layer

Now that the Concept Bottleneck Layer is learned, the next task is to learn the final predictor with the fully connected layer $W_F \in \mathbb{R}^{d_z \times M}$ where d_z is the number of output classes. The goal is to keep W_F sparse, since sparse layers have been demonstrated to be more interpretable [103]. Given that both the backbone $f(x)$ and learned concept projection W_c are fixed, this is a problem of learning a sparse linear model, which can be solved efficiently with the

elastic net objective:

$$\min_{W_F, b_F} \sum_{i=1}^N L_{ce}(W_F f_c(x_i) + b_F, y_i) + \lambda R_\alpha(W_F) \quad (5.2)$$

where $R_\alpha(W_F) = (1 - \alpha) \frac{1}{2} \|W_F\|_F^2 + \alpha \|W_F\|_{1,1}$, $\|\cdot\|_F$ denotes the Frobenius norm, $\|\cdot\|_{1,1}$ denotes element-wise matrix norm, b_F denotes the bias of the FC layer, L_{ce} is the standard cross-entropy loss and y_i is the ground-truth label of data x_i . We optimize Equation (5.2) using the GLM-SAGA solver created by [103]. For the sparse models, we used $\alpha = 0.99$ and λ was chosen such that each model has 25 to 35 nonzero weights per output class. This level was found to still result in interpretable decisions while retaining good accuracy. Depending on the number of features/concepts in the previous layer this corresponds to 0.7-15% of the weights of the model being nonzero.

5.4 Experiment Results

We present three main results on evaluating the accuracy and interpretability of the Label-free CBM in this section. Due to page limit, additional experiments and discussions are in Appendix E.1.2-E.1.12. An overview of additional experiments is provided in Appendix E.1.1.

Datasets. To evaluate our approach, we train Label-free CBMs on 5 datasets. These are CIFAR-10, CIFAR-100 [52], CUB [98], Places365 [115] and ImageNet [19]. This is a diverse set of tasks, where CIFAR-10/100 and ImageNet are general image classification datasets, CUB is a fine-grained bird-species classification dataset and Places365 is focused on scene recognition. Their sizes vary greatly, with CUB having 5900 training samples, CIFAR datasets 50,000 each and ImageNet and Places365 have 1-2 million training images. CUB contains annotations with 312 concepts for each datapoint, such as *has wing color:blue* or *has head pattern:spotted*. We used neither the concept names or the concept annotations to train our LF-CBM models to showcase our ability to perform without labels, yet our method discovered similar concepts and is competitive with methods that utilize the available concept information.

Setup. For CIFAR and CUB we use the same backbone models as [108] for fair

comparison, so we use CLIP(RN50) image encoder as the backbone for CIFAR, and ResNet-18 trained on CUB from imgclsmob for CUB. For both ImageNet and Places365 we use ResNet-50 trained on ImageNet as the backbone network. The number of concepts each model uses is roughly proportional to the number of output classes for that task, as each class adds more initial concepts. The number of concepts for our models are as follows: 128 for CIFAR-10, 824 for CIFAR-100, 211 for CUB, 2202 for Places-365 and 4505 for ImageNet. CUB has a smaller number of concepts because we only used the *important features* prompt. All models are trained on a single Nvidia Tesla P100 GPU, and the full training run takes anywhere from few minutes to 20 hours depending on the dataset size. The majority of runtime is taken by step 2, where we save activations for both the backbone and CLIP over the entire training dataset. Fortunately, these results only need to be calculated once and can be reused for training new CBMs on the same dataset. Once the activations have been saved, learning the model takes less than 4 hours on all datasets.

Result (I): Accuracy. Table 5.2 shows the performance of Label-free CBM on all 5 datasets. We can see that our method can create a CBM with a sparse final layer with little loss in accuracy on all datasets, including a model with 72% top-1 accuracy on ImageNet. Our Label-free CBM has significantly higher accuracy than Post-hoc CBM on the datasets we evaluated, but some rows for P-CBM are N/A as they do not provide those results and it is unclear how to scale P-CBM to larger datasets. For Table 5.2 we excluded methods with non-interpretable components (i.e. P-CBM-h or IBD) as we want to focus on fully interpretable CBM models. Note that P-CBM uses the expert provided concept set for CUB200, yet we can outperform it using our fully GPT-3 derived set of features. The standard sparse models were finetuned by us by learning a sparse final layer directly after feature layer $f(x)$ as described in [103] and also have 25-35 nonzero weights per class. The accuracies of sparse standard models are comparable to our CBM, indicating the CBM does not reduce accuracy.

Result (II): Explainable decision rules. Perhaps the biggest benefit of Concept Bottleneck Models is that their decisions can be explained as a simple linear combination of

Table 5.2. Accuracy comparison, best performing sparse model bolded. We can see our method outperforms Posthoc-CBM and performs similarly to a sparse standard model. The results for our method are mean and standard deviation over three training runs. *Indicates reported accuracy.

Model	Sparse final layer	Dataset				
		CIFAR10	CIFAR100	CUB200	Places365	ImageNet
Standard	No	88.80%*	70.10%*	76.70%	48.56%	76.13%
Standard (sparse)	Yes	82.96%	58.34%	75.96%	38.46%	74.35%
P-CBM	Yes	70.50%*	43.20%*	59.60%*	N/A	N/A
P-CBM (CLIP)	Yes	84.50%*	56.00%*	N/A	N/A	N/A
Label-free CBM (Ours)	Yes	86.40% $\pm 0.06\%$	65.13% $\pm 0.12\%$	74.31% $\pm 0.29\%$	43.68% $\pm 0.10\%$	71.95% $\pm 0.05\%$

understandable features. To showcase this, in Figure 5.3 we visualize the final layer weights for the classes "Orange" and "Lemon" on ImageNet, and classes "Mountain" and "Mountain Snowy" on Places365. This visualization is a Sankey diagram of the final layer weights coming into two output classes, with the weight between a concept and output class displayed as the width of the line connecting them. We have only included weights with absolute value greater than 0.05 (for comparison the largest weights are usually 0.5-1). Negative weights are denoted as NOT concept. The learned decision rules largely align with our intuitions. For ImageNet, the concept "citrus" fruit is highly connected to both "Orange" and "Lemon", while orange colors activate the "Orange" class and yellow colors and limes activate the "Lemon" class. On Places, concepts like "a high elevation" activate both classes, while "Mountain Snowy" is activated by snow and ice related concepts and "Mountain" is activated by concepts related to volcanoes like "lava" and "crater". Visualizations like this allow us to gain a *global* understanding of how our model behaves. Additional visualizations are shown in Appendix E.1.11.

Result (III): Explainable individual decisions.

In addition to global understanding, CBMs allow us to understand the reasoning behind individual decisions. Since our decisions are linear functions of interpretable features, they can be explained in a simple and accurate way. After learning the Concept Bottleneck Layer, we normalize the activations of each concept to have mean 0 and standard deviation 1 on the

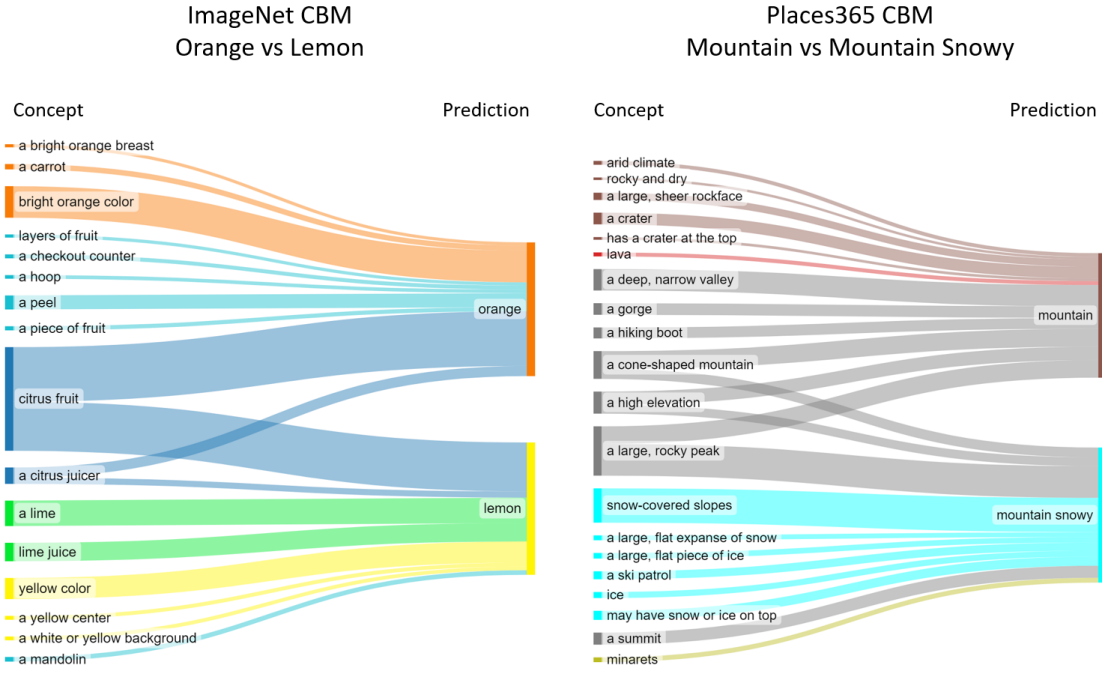


Figure 5.3. Visualization of the final layer weights of our Label-free CBM. Showcasing how our models differentiate between two similar classes.

training data. Given normalized features, the contribution of feature j to output i on input x_k can be naturally calculated as $Contrib(x_k, i, j) = W_{F[i,j]} f_c(x_k)_j$. Since our W_F is sparse, most contributions will be 0 and the important contributions can be easily visualized with a bar plot. Example visualizations for explaining the model’s predicted class are shown in Figure 5.4, and more visualizations are available in Appendix E.1.12. These visualizations show the concepts with highest absolute value contribution. Concepts with negative activation can still be important contributors, and they are shown as “NOT concept” in our visualizations. We can see for example the CUB model correctly identifies “a red head” as the most important feature in recognizing this image as “Red headed Woodpecker”, while the fact that there is no “a rosy breast” present also increases model’s confidence.

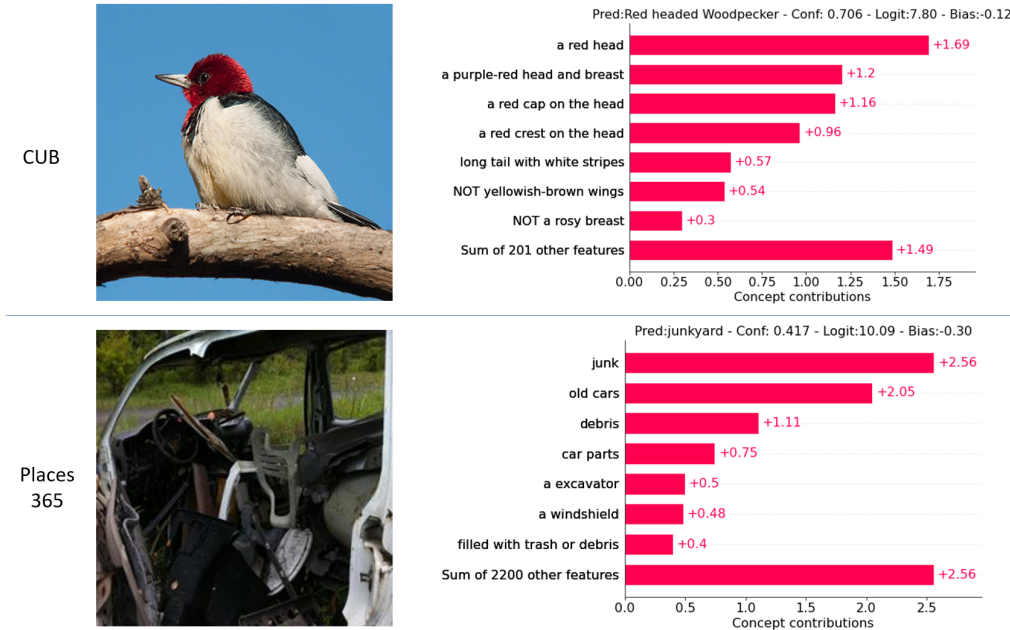


Figure 5.4. Visualization of two correct decisions made by our Label-free CBM.

5.5 Case study: Manually improving an ImageNet model

In this section we take a deep dive into our ImageNet Label-free CBM – we analyze the errors it makes and design ways to debug/improve our network based on these findings. In particular, we show how we can inspect a handful of individual incorrect decisions and manually change our model weights to not only fix those predictions but also multiple other predictions, improving the network’s overall accuracy. To our knowledge, this is the first example of manually editing a large well-trained neural network in a way that improves test accuracy on the same distribution.

5.5.1 Types of model errors

During the course of our experiment, we were able to identify 4 different types of mistakes our model makes, with different strategies for fixing them:

Type 1: Incorrect/ambiguous label: Despite looking like errors in terms of accuracy, these are simply a consequence of having noisy labels and are unavoidable. We will not attempt

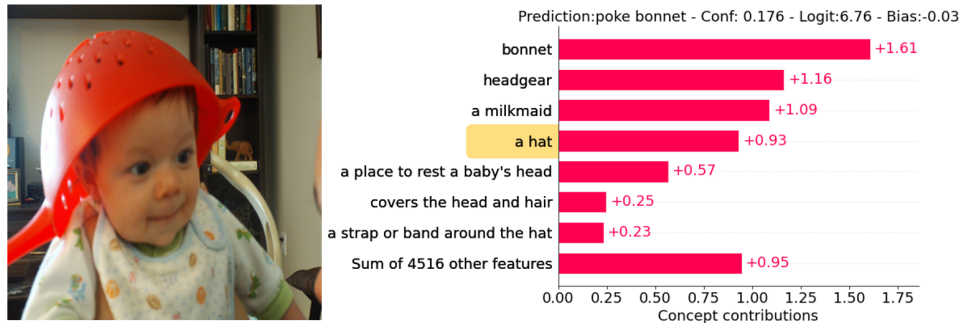
to fix this type of mistake.

Type 2: No sufficient concept in CBL: Despite having a large set of concepts, our CBL does not include all concepts important for detecting certain classes. For example, detecting the difference between two species of snakes may require recognizing very fine-grained patterns that can't be easily explained in words. While these could often be fixed by adding new concepts to the concept set, we do not focus on it as it requires retraining the projection W_c and the final layer.

Type 3: Incorrect concept activations: Some of the time, the activations in our CBL are incorrect and do not match the image, causing the prediction to become incorrect. Since the network up to the Concept Bottleneck Layer is mostly a black box, we can't easily improve the predictions, but predictions can be improved with test time interventions as shown in Figure 5.5.

Type 4: Incorrect final layer weight W_F : Sometimes even if all the concept activations are correct, the model's final layer weights still cause it to make an erroneous prediction. As the final layer is fully interpretable, fixing these errors is the main focus of our study described in Sec 5.5.2.

Examples of Type 1 and Type 2 errors are shown in Figure E.7 in the Appendix.



Intervene: activation of "a hat" 2.70 -> 0. New prediction: **strainer** ✓

Figure 5.5. Fixing an incorrect model prediction on ImageNet by simply zeroing out the incorrect activation for concept "a hat". Model originally predicted "poke bonnet" which is a type of hat.

5.5.2 Editing final layer weights

In this section we describe our procedure for manually editing final layer weights of our model. This is done with the following procedure:

1. **Find an input where the model makes a Type 4 error:** This is done by visualizing incorrect model predictions and their explanations, and identifying Type 4 errors, i.e. errors where the highly activating concepts look correct and the label is correct and unambiguous.
2. **Identify a concept to edit.:** Perhaps the most tricky step, to change a prediction we need to select a concept (using our domain knowledge) that is highly activated in this image, important for the ground truth class, not important for the incorrect class, and will have a small impact on other predictions. For example to flip a prediction from "cricket insect" to "grasshopper" we chose the concept "a green color".
3. **Change weights by sufficient magnitude:** Next we choose magnitude of weight change Δw , and edit the weights in the following way: $W_{F[gt,concept]} \leftarrow W_{F[gt,concept]} + \Delta w$ and $W_{F[pred,concept]} \leftarrow W_{F[pred,concept]} - \Delta w$. Typically we want Δw to be slightly more than required to flip the prediction on our example, but understanding its effect on other predictions is hard and should be checked with a validation dataset.

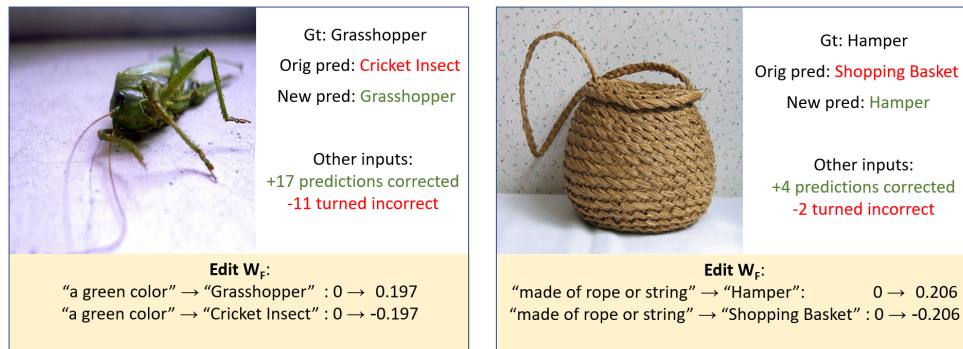


Figure 5.6. Examples of two edits we performed on our model predictions. Both edits fix the error we found, and fix more other predictions than they cause mistakes on other unseen inputs.

Fig 5.6 shows two examples of model edits we performed, and their effects on model

predictions on the validation set. In total we identified 5 such beneficial edits during our quick exploration, the other 3 are shown in Fig E.6. After applying all of them, the total validation accuracy of our model goes from 71.98% to 72.02%, correcting 38 predictions while turning 17 incorrect. While this might seem like a small difference, it is worth noting we only change 10 out of 4.5 million total weights in the final layer(of which 34,550 are non-zero). Our edits only affect 10 classes, which have a total of 500 validation examples, so in effect we increase accuracy on this subset by 4.2% which is a significant boost. We believe this editing approach is a promising direction for future use, where practitioners could for example notice an incorrect prediction in a production system, quickly devise a fix for that prediction, and check how it affects other predictions using validation data.

5.6 Conclusion

We have presented Label-free CBM, a fully automated and scalable method for generating concept bottleneck models and used it to create the first high performing CBM on ImageNet. In addition, we have demonstrated how our models are easier to understand in terms of both global decision rules and reasoning for individual predictions. Finally, we show how to use this understanding to manually edit weights to improve accuracy in a trained ImageNet model.

Acknowledgements

The authors would like to thank anonymous reviewers for valuable feedback to improve the manuscript. The authors also thank MIT-IBM Watson AI lab for support in this work. This work was done during T. Oikarinen’s internship at MIT-IBM Watson AI Lab. T.-W. Weng is supported by National Science Foundation under Grant No. 2107189.

Chapter 5, in full, is based on the paper from Tuomas Oikarinen, Subhro Das, Lam M Nguyen and Tsui-Wei Weng, ”Label-free Concept Bottleneck Models” published at the International Conference on Learning Representations (ICLR), 2023. The dissertation author is

the primary investigator and author of this paper.

Appendix A

Supplementary Material for Chapter 1

A.1 Additional Results and Details

A.1.1 Similarity function details and derivation

Rank reorder hyperparameters:

The results of Table 1.3 are using top 5% of most highly activating images for each neuron and using $p = 3$ for the l_p -norm.

WPMI:

In this section, we show that one choice of similarity function $\text{sim}(t_m, q_k; P)$ can be derived based on the weighted point-wise mutual information (wpmi). Note that wpmi is also used in [39] but in a different way – our approach can compute wpmi directly from the CLIP products P and does not require any training, while [39] train two models to estimate wpmi.

To start with, by definition, the wpmi between a concept t_m and the most highly activated images B_k of neuron k can be written as

$$\text{wpmi}(t_m, q_k) = \log p(t_m | B_k) - \lambda \log p(t_m) \quad (\text{A.1})$$

Here B_k is the set of images that most highly activates neuron k , i.e. the top indices of q_k . First we can compute $p(t_m | x_i) = \text{softmax}(aP_{i,:})_m$, where $\text{softmax}(z)_n = \frac{e^{z_n}}{\sum_{j=1}^N e^{z_j}}$ with $z \in \mathbb{R}^N$, $P_{i,:}$ is the i -th row vector of the concept-activation matrix P and a is a scalar temperature constant. This is

the probability that CLIP assigns to a concept t_m for image x_i when used as a classifier.

We then define $p(t_m|B_k)$ as the probability that all images in B_k have the concept t_m , which gives us $p(t_m|B_k) = \prod_{x_i \in B_k} p(t_m|x_i)$. Thus, we have

$$\log p(t_m|B_k) = \sum_{x_i \in B_k} \log p(t_m|x_i) \quad (\text{A.2})$$

which is the 1st term in Eq (A.1). Next, we can approximate the 2nd term $p(t_m)$ in Eq (A.1) as follows: $p(t_m)$ is the probability that a random set of images B will be described by t_m . Since we don't know the true distribution for a set of images, an efficient way to approximate this is to average the probability of t_m over the different neurons we are probing. This can be described by the following equation:

$$p(t_m) = \mathbb{E}_B[p(t_m|B)] \approx \frac{\sum_{j \in C} p(t_m|B_j)}{|C|} = \frac{\sum_{j \in C} \prod_{x_i \in B_j} p(t_m|x_i)}{|C|} \quad (\text{A.3})$$

where C is the set of neurons in the layer we are probing. Thus we can plug Eq. (A.2) and Eq. (A.3) into Eq. (A.1) to compute wpmi through the CLIP model:

$$\text{wpmi}(t_m, q_k) = \sum_{x_i \in B_k} \log p(t_m|x_i) - \lambda \log \left(\sum_{j \in C} \prod_{x_i \in B_j} p(t_m|x_i) \right) + \lambda \log |C| \quad (\text{A.4})$$

So we can use the above Eq (A.4) in our CLIP-Dissect and set $\text{sim}(t_m, q_k; P) = \text{wpmi}(t_m, q_k)$ in the algorithm.

For our experiments we use $a = 2$, $\lambda = 0.6$ and top 28 most highly activating images for neuron k as B_k which were found to give best quantitative results when describing final layer neurons of ResNet-50.

SoftWPMI:

SoftWPMI is an extension of wpmi as defined by Eq. (A.4) into settings where we have

uncertainty over which images should be included in the example set B_k . In WPMI the size of example set is defined beforehand, but it is not clear how many images should be included, and this could vary from neuron to neuron. In this description, we assume that there exists a true B_k which includes images from D_{probe} if and only if they represent the concept of neuron k . We then define binary indicator random variables $X_i^k = \mathbb{1}[x_i \in B_k]$ which take value 1 if the i th image is in set the B_k , and we define $X^k = \{X_1^k, \dots, X_M^k\}$.

Our derivation begins from the observation that we can rewrite $p(t_m|B_k)$ from above as:

$$p(t_m|B_k) = \prod_{x_i \in B_k} p(t_m|x_i) = \prod_{x_i \in D_{probe}} p(t_m|x_i)^{\mathbb{1}[x_i \in B_k]} = \prod_{x_i \in D_{probe}} p(t_m|x_i)^{X_i^k} \quad (\text{A.5})$$

Now:

$$\mathbb{E}_{X_i^k}[p(t_m|x_i)^{X_i^k}] = p(x_i \in B_k)p(t_m|x_i) + (1 - p(x_i \in B_k)) = 1 + p(x_i \in B_k)(p(t_m|x_i) - 1) \quad (\text{A.6})$$

If we assume the X_i^k are statistically independent, we can write:

$$\mathbb{E}_{X^k}[p(t_m|B_k)] = \prod_{x_i \in D_{probe}} \mathbb{E}_{X_i^k}[p(t_m|x_i)^{X_i^k}] = \prod_{x_i \in D_{probe}} [1 + p(x_i \in B_k)(p(t_m|x_i) - 1)] \quad (\text{A.7})$$

$$\Rightarrow \log \mathbb{E}_{X^k}[p(t_m|B_k)] = \sum_{x_i \in D_{probe}} \log(1 + p(x_i \in B_k)(p(t_m|x_i) - 1)) \quad (\text{A.8})$$

Note Equation (A.6) goes to 1 if $p(x_i \in B_k) = 0$ (i.e. no effect in a product) and to $p(t_m|x_i)$ if $p(x_i \in B_k) = 1$. So Eq. (A.8) reduces to Eq. (A.2) of standard WPMI if $p(x_i \in B_k)$ is either 1 or 0 for all $x_i \in D_{probe}$. In other words, we are considering a "soft" membership in B_k instead of "hard" membership of standard WPMI.

To get the second term for wpmi, $p(t_m)$, i.e. probability that text t_m describes a random

example set B_k , we can approximate it like we did in Eq. (A.3) by using the example sets for other neurons we are interested in.

$$\begin{aligned}
p(t_m) &= \mathbb{E}_{B_i}[\mathbb{E}_{X^i}[p(t_m|B_i)]] \approx \frac{\sum_{j \in C} \mathbb{E}_{X^j}[p(t_m|B_j)]}{|C|} \\
&\rightarrow \frac{\sum_{j \in C} \mathbb{E}_{X^j}[p(t_m|B_j)]}{|C|} = \frac{\sum_{j \in C} \Pi_{x \in D_{probe}} [1 + p(x \in B_j)(p(t_m|x) - 1)]}{|C|} \quad (\text{A.9})
\end{aligned}$$

Finally, we can compute full SoftWPMI with Eq. (A.8) and Eq. (A.9) and use it as similarity function in CLIP-Dissect:

$$\begin{aligned}
\text{soft_wpmi}(t_m, q_k) &= \sum_{x_i \in D_{probe}} \log(1 + p(x_i \in B_k)(p(t_m|x_i) - 1)) \\
&\quad - \lambda \log \left(\sum_{j \in C} \Pi_{x \in D_{probe}} [1 + p(x \in B_j)(p(t_m|x) - 1)] \right) + \lambda \log |C| \quad (\text{A.10})
\end{aligned}$$

One thing we haven't yet discussed is the choice of $p(x \in B_k)$. There is flexibility and this probability could be derived from the activations of neuron k on image x , by for example by taking a scaled sigmoid, or it could be based on the ranking of the image.

For our experiments we found ranking based probability to perform the best, and used $p(x \in B_k)$ linearly decreasing from 0.998 of the most highly activating image for neuron k to 0.97 for 100th most highly activating image and 0 for all other images. Thus in practice we only have to use the 100 images when calculating SoftWPMI instead of full D_{probe} which is much more computationally efficient. For other hyperparameters we used $a = 10$ and $\lambda = 1$.

A.1.2 Additional qualitative results

Additional visualization on ResNet-18 and ResNet-50 in Figs A.1 and A.2, continued from Section 1.4.1. We have subjectively colored the descriptions green if they match these 5 images, yellow if they match but are too generic and red if they do not match. In Fig. A.1 we

can see both Network Dissection and CLIP-Dissect do very well while MILAN struggles to explain some neurons. MILAN(b) is trained on both ImageNet and Places365 networks, while MILAN(i) is only trained on ImageNet. Both MILAN networks perform similarly here but the ImageNet version misses/is too generic for more neurons, such as labeling a bus neuron as "vehicles". The neurons on the left have highest IoU according to MILAN while neurons on the right have highest similarity to the concept according to our similarity function. In Fig. A.2 we see that both CLIP-Dissect and Network Dissection perform well on these most interpretable neurons except for a few failures by Network Dissection, while MILAN often gives concepts that are too generic. MILAN(b) is trained on both ImageNet and Places365 networks, while MILAN(p) is only trained on Places365. We can see the Places trained model is struggling more with concepts like spiders, indicating issues with generalization.

Resnet-18(Places 365) Layer 4

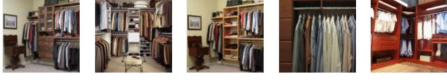
Most interpretable neurons (Network Dissection)	Most interpretable neurons (CLIP-Dissect)
Neuron 453: CLIP-Dissect: chevrolet MILAN(b): Vehicles 	Neuron 432: CLIP-Dissect: attic MILAN(b): Rooms 
Neuron 167: CLIP-Dissect: trains MILAN(b): Train track and train tracks 	Neuron 88: CLIP-Dissect: fountains MILAN(b): Water fountains 
Neuron 291: CLIP-Dissect: puppies MILAN(b): Dog heads 	Neuron 189: CLIP-Dissect: attic MILAN(b): Buildings 
Neuron 384: CLIP-Dissect: closet MILAN(b): Closets 	Neuron 17: CLIP-Dissect: auditorium MILAN(b): Chairs 
Neuron 383: CLIP-Dissect: bike MILAN(b): Wheels of vehicles 	Neuron 350: CLIP-Dissect: bathroom MILAN(b): Furniture 
Neuron 109: CLIP-Dissect: exhibitions MILAN(b): Walls 	Neuron 476: CLIP-Dissect: workstations MILAN(b): Furniture 
Neuron 385: CLIP-Dissect: buses MILAN(b): Buses 	Neuron 456: CLIP-Dissect: kitchens MILAN(b): Circular objects 
Neuron 256: CLIP-Dissect: buses MILAN(b): Buses 	Neuron 3: CLIP-Dissect: auditorium MILAN(b): Green colored objects 
Neuron 483: CLIP-Dissect: kayaking MILAN(b): Water 	Neuron 285: CLIP-Dissect: attic MILAN(b): Buildings 
Neuron 192: CLIP-Dissect: towers MILAN(b): Towers 	Neuron 413: CLIP-Dissect: mosque MILAN(b): Towers 

Figure A.1. Explanations of most interpretable neurons in the second to last layer of ResNet-18 trained on Places365. Displayed together with 5 most highly activating images for that neuron.

Resnet-50(ImageNet) Layer 4

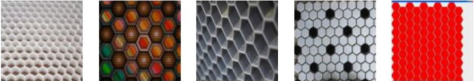
Most interpretable neurons (Network Dissection)	Most interpretable neurons (CLIP-Dissect)
Neuron 39: CLIP-Dissect: cat MILAN(b): Animals 	Neuron 720: CLIP-Dissect: kitchens MILAN(b): Furniture and electronics 
Neuron 465: CLIP-Dissect: webs MILAN(b): Items with straight features 	Neuron 516: CLIP-Dissect: bookshop MILAN(b): Shelves and books 
Neuron 632: CLIP-Dissect: lattice MILAN(b): Soccer balls 	Neuron 210: CLIP-Dissect: collier MILAN(b): Items that are connected 
Neuron 76: CLIP-Dissect: web MILAN(b): Insects, spiders 	Neuron 972: CLIP-Dissect: gymnastics MILAN(b): People 
Neuron 723: CLIP-Dissect: crochet MILAN(b): Fabric 	Neuron 736: CLIP-Dissect: crab MILAN(b): Animals and food 
Neuron 1966: CLIP-Dissect: knitting MILAN(b): Similar color patterns 	Neuron 1517: CLIP-Dissect: bookshop MILAN(b): Books 
Neuron 1742: CLIP-Dissect: lattice MILAN(b): Items with circular features 	Neuron 1108: CLIP-Dissect: broccoli MILAN(b): Green objects 
Neuron 1959: CLIP-Dissect: spider MILAN(b): Bugs 	Neuron 76: CLIP-Dissect: web MILAN(b): Insects, spiders 
Neuron 1899: CLIP-Dissect: lattice MILAN(b): Rounded edges in pictures 	Neuron 1529: CLIP-Dissect: kitchens MILAN(b): Food items and people 
Neuron 393: CLIP-Dissect: lattice MILAN(b): Rounded objects 	Neuron 723: CLIP-Dissect: crochet MILAN(b): Fabric 

Figure A.2. Explanations of most interpretable neurons in the second to last layer of ResNet-50 trained on ImageNet. Displayed together with 5 most highly activating images for that neuron.

A.1.3 Low level concepts

In this section we show additional results of probing low level concepts, using two networks trained on ImageNet, ResNet-50 and ResNet-152. For ResNet-152 we also compare against human annotations for these neurons from MILAnnotations [39].

The results for ResNet-152 can be seen in Figure A.3. We can see CLIP-Dissect is able to accurately detect many lower level concepts, such as colors in Conv1 neurons 1,3,10 and Layer1 neuron 4, as well as detecting that neuron 3 of Layer1 activates specifically for the *text/label*, without having access to the activation pattern, while MILAN fails to detect this.

However, CLIP-Dissect does also have some failure modes on lower level patterns.

- **Failure to differentiate between concept and correlated objects:** This leads to higher level outputs than desired. For example: CLIP-Dissect gives the concept *underwater* to conv1 neuron 16, while the true concept is probably more similar to *blue* (human annotators also made this mistake), or conv1 neuron 28 where CLIP-Dissect outputs *zebra* while the neuron is likely just detecting *stripes*. The worst example of this is neuron conv1 neuron 24, which simply activates on white background, but this is entirely missed by CLIP-Dissect as it's not good at focusing on the background.
- **Uninterpretable neurons:** Some neurons seem to be not interpretable, e.g. Conv1 neuron 2. CLIP-Dissect outputs *music* which seems incorrect, but neither human annotators or MILAN were able to assign a clear concept for the neuron either. See Appendix A.1.8 for more analysis on uninterpretable neurons.

Similar observations also hold for the ResNet-50 model as can be seen in Figure A.4. Note we do not have human annotations to compare against for this network.

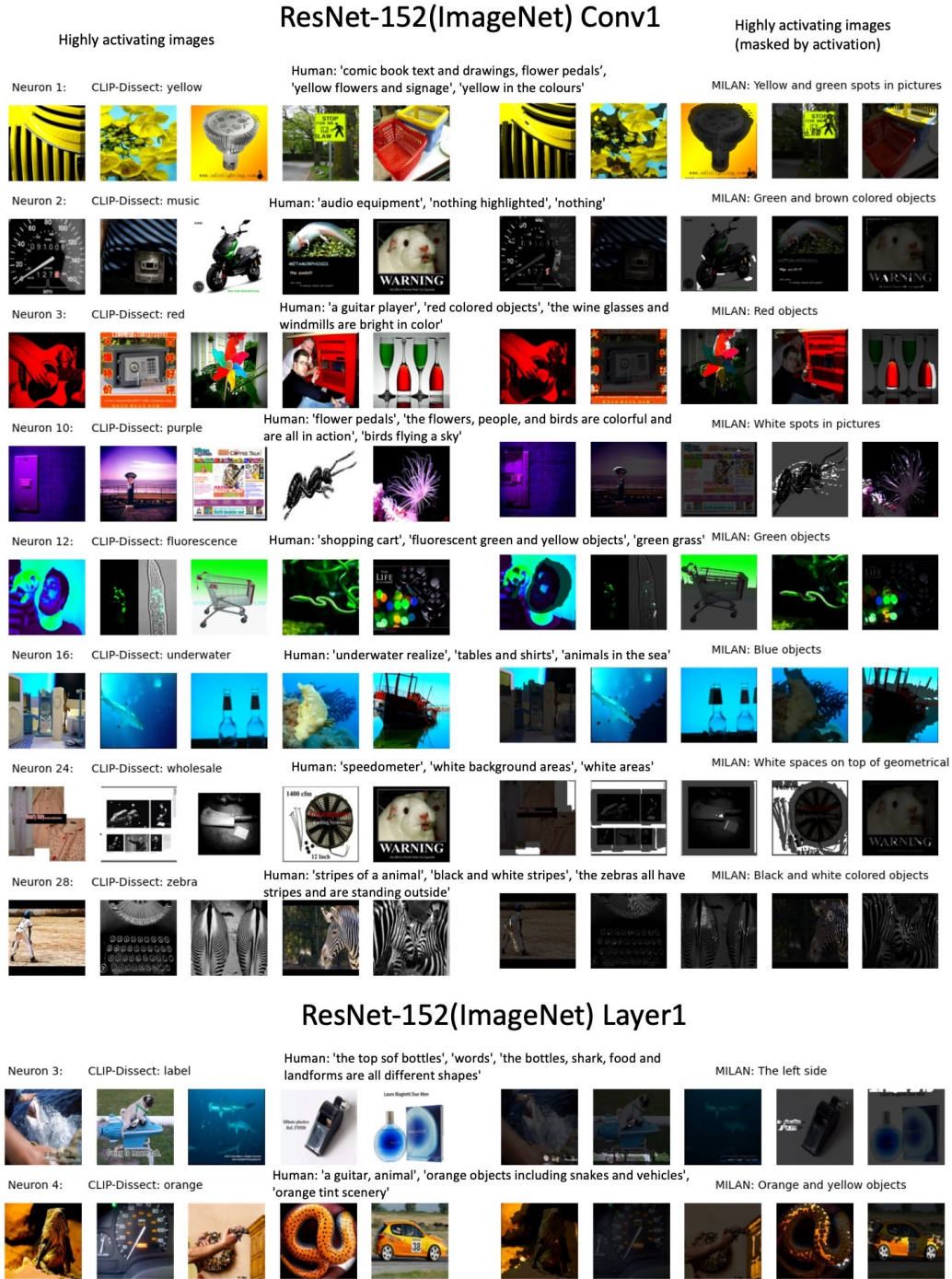


Figure A.3. Descriptions for select neurons in early layers of ResNet-152, showcasing both the successes and failure modes of CLIP-Dissect. For this figure we used max as the summary function g for CLIP-Dissect to be comparable to MILAnnotations. For evaluation we used the MILAN model trained on only Places models to avoid overfitting.

ResNet-50 (ImageNet) Layer1

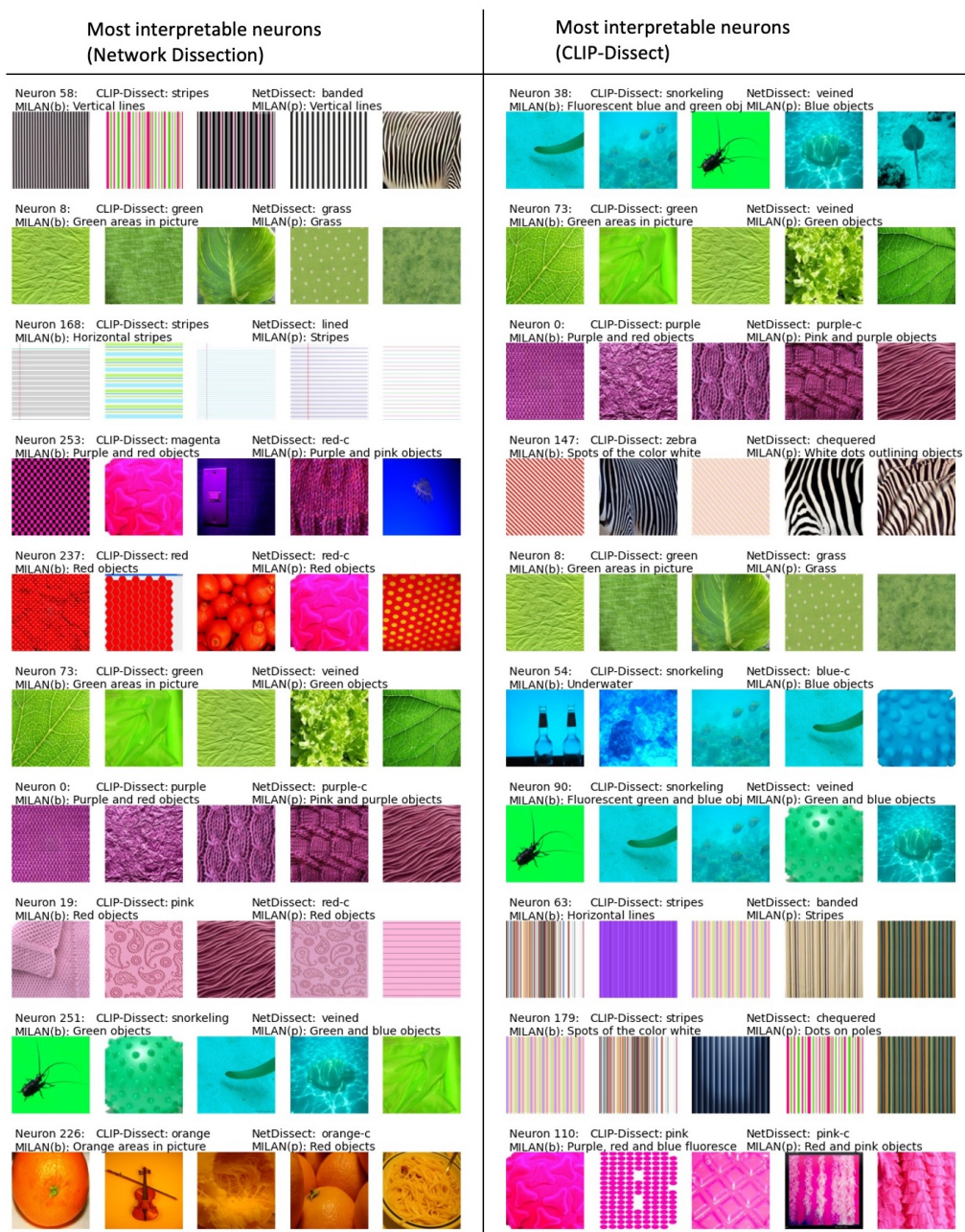


Figure A.4. Descriptions of most interpretable neurons (highest similarity/IoU) of an early layer in ResNet-50.

A.1.4 Compositional Concepts

In the sections above our method has focused on choosing the most fitting concept from the pre-defined concept set. While changing the concept set in CLIP-Dissect is as easy as editing a text file, we show it can also detect more complex compositional concepts. We experimented with generating explanations by searching over text concatenations of two concepts in our concept space. To reduce computational constraints, we only looked at combinations of 100 most accurate single word labels for each neuron. Example results are shown in Fig A.5. While the initial results are promising, some challenges remain to make these compositional explanations more computationally efficient and consistent, which is an important direction for future work.

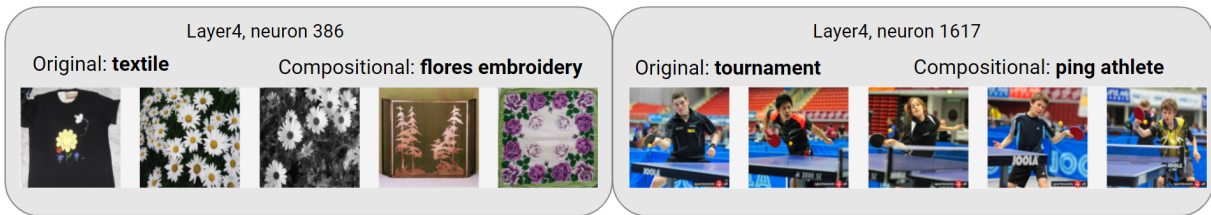


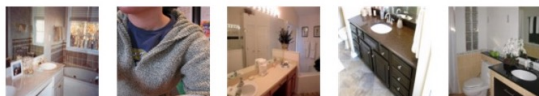
Figure A.5. An example of compositional explanations generated by our method for two neurons of ResNet50 trained on ImageNet.

A.1.5 Vision Transformer

Since our method does not rely on the specifics of CNNs in its operation, we can easily extend it to work on different architectures, such as the Vision Transformer, specifically ViT-B/16 [20] model trained on ImageNet. We have visualized most interpretable neurons, their highly activating images and their descriptions in Figure A.6. Interestingly, we found the highly interpretable neurons to be very location focused, i.e. *kitchens* or *highways* despite the network being trained on object level labels (ImageNet).

ViT-B/16(ImageNet) Most interpretable neurons

Neuron 665: CLIP-Dissect: bathroom



Neuron 82: CLIP-Dissect: kitchens



Neuron 507: CLIP-Dissect: kitchens



Neuron 185: CLIP-Dissect: kitchens



Neuron 725: CLIP-Dissect: highways



Neuron 631: CLIP-Dissect: bathroom



Neuron 727: CLIP-Dissect: kitchen



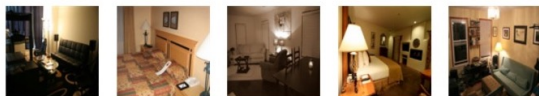
Neuron 410: CLIP-Dissect: kitchen



Neuron 317: CLIP-Dissect: bedrooms



Neuron 363: CLIP-Dissect: room



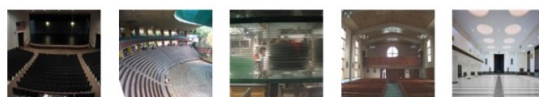
Neuron 122: CLIP-Dissect: fields



Neuron 746: CLIP-Dissect: bedrooms



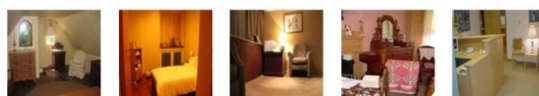
Neuron 574: CLIP-Dissect: auditorium



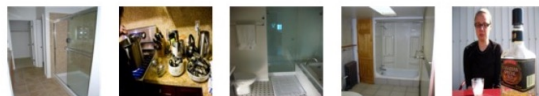
Neuron 226: CLIP-Dissect: furnished



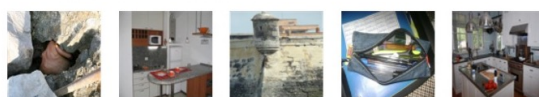
Neuron 752: CLIP-Dissect: room



Neuron 522: CLIP-Dissect: bathroom



Neuron 625: CLIP-Dissect: kitchens



Neuron 236: CLIP-Dissect: kitchen



Neuron 325: CLIP-Dissect: kitchen



Neuron 457: CLIP-Dissect: puppy

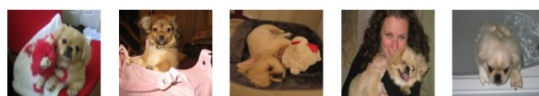


Figure A.6. Most interpretable neurons after the 'encoder' module (one of the last layers) of a Vision Transformer.

A.1.6 Predicting input class from neuron descriptions

In this section we follow reviewer suggestion and study whether the class of the image can be predicted based on descriptions of highly activating neurons. We perform an experiment on the neurons in the second-to-last layer (last hidden layer) and study if the descriptions of highly activating neurons in this layer can be used to predict what class the input is from.

As we show in below experiments, the neuron description generated from CLIP-Dissect gives overall higher prediction accuracy than the neuron descriptions from prior work (Network Dissection and MILAN), which suggests our proposed method generates better neuron description than prior work. Below we outline the steps we used to predict the class of an image based on the internal neuron descriptions from different neuron labelling methods:

1. Following our notation in Sec 1.3, for an image x_i , record the activations of neurons in the second-to-last layer $g(A_j(x_i))$ (where j denotes the neuron index) as well as the predicted class c .
2. Find the neuron with the highest positive contribution to the predicted class c , which can be computed as $k = \arg \max_j W_{c,j} \cdot g(A_j(x_i))$.
3. Obtain the description t_k for neuron k using an automated description method.
4. Find the class name that is most similar to the description of the highest contributing neuron, and predict this class as the images class. Similarity is measured using an average of cosine similarities in the sentence embedding space of CLIP text encoder and mpnet sentence embedding space discussed in section 1.4.2.

We performed this experiment on ResNet-50 trained on ImageNet, and found that predicting with the above algorithm and CLIP-Dissect neuron descriptions we were able to correctly predict the class of 10.28% of the images. In contrast, when we used descriptions from Network Dissection we were only able to predict the class of 3.36% of the images, and with MILAN(base)

only 2.31% of the time. This gives evidence towards our CLIP-Dissect descriptions being higher quality than Network Dissection and MILAN. It’s worth noting that overall we would not expect to reach a very high predictive accuracy using the above method as the most contributing neuron often does have a completely different role than the target class. However if we study the same network with different neuron description methods, we would expect a better neuron description method would return a higher prediction accuracy, assuming at least some of the important neurons indeed do have similar role to the class itself. This method has the benefit of being automated, going beyond visualizing few most highly activating images and being able to analyze hidden layer description quality. We think methods like this are an interesting future direction for evaluating neuron description methods.

A.1.7 Visualizing wider range of activations

So far our qualitative evaluation has focused on whether the description matches the 5 or 10 images in $\mathcal{D}_{\text{probe}}$ that activate the neuron the most. However this does not give a full picture of the function of the neuron, and in this section we explore how the neurons activate on a wider range of input images. In particular, in Figures A.7 and A.8 we visualize the most interpretable neurons of two layers of ResNet-18 (Places 365) and ResNet-50 (ImageNet) by uniformly sampling images from the top 0.1%, 1% 5% of most highly activating images. We can see that the descriptions tend to match quite well for the 0.1% of most highly activating images, but the top 1% and top 5% images start to be of quite different concepts only slightly related to the description. We also notice that low level concepts like colors tend to be more consistently represented in top 1% and 5% images while higher level concepts are not. This result is somewhat expected as the $\mathcal{D}_{\text{probe}}$ does not have that many images for each higher level concept, but highlights the need to explore neuron activations beyond just the few most highly activated images.

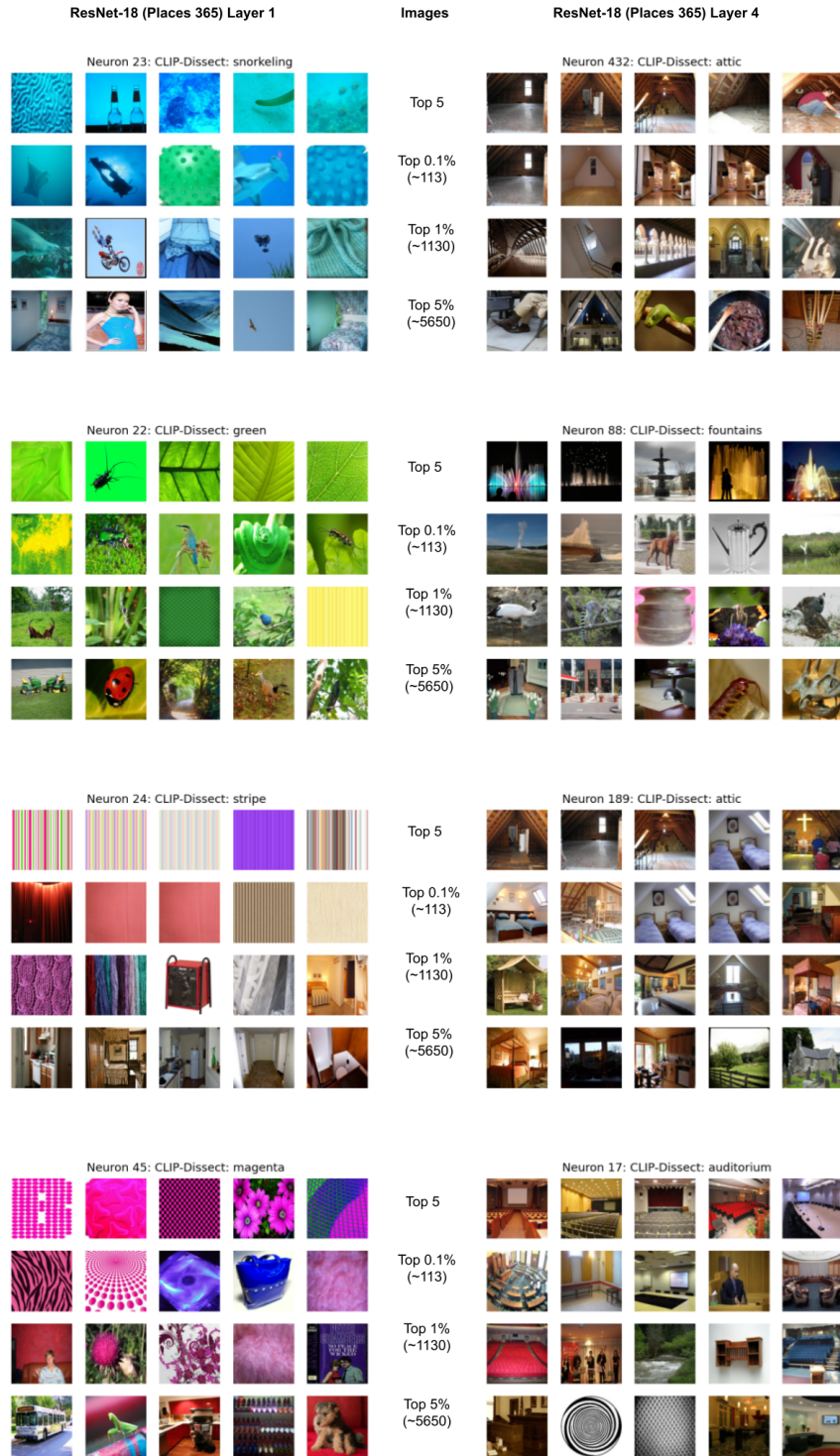


Figure A.7. Most interpretable neurons of ResNet-18, showcasing randomly sampled images of a wide range of most highly activating images for that neuron.

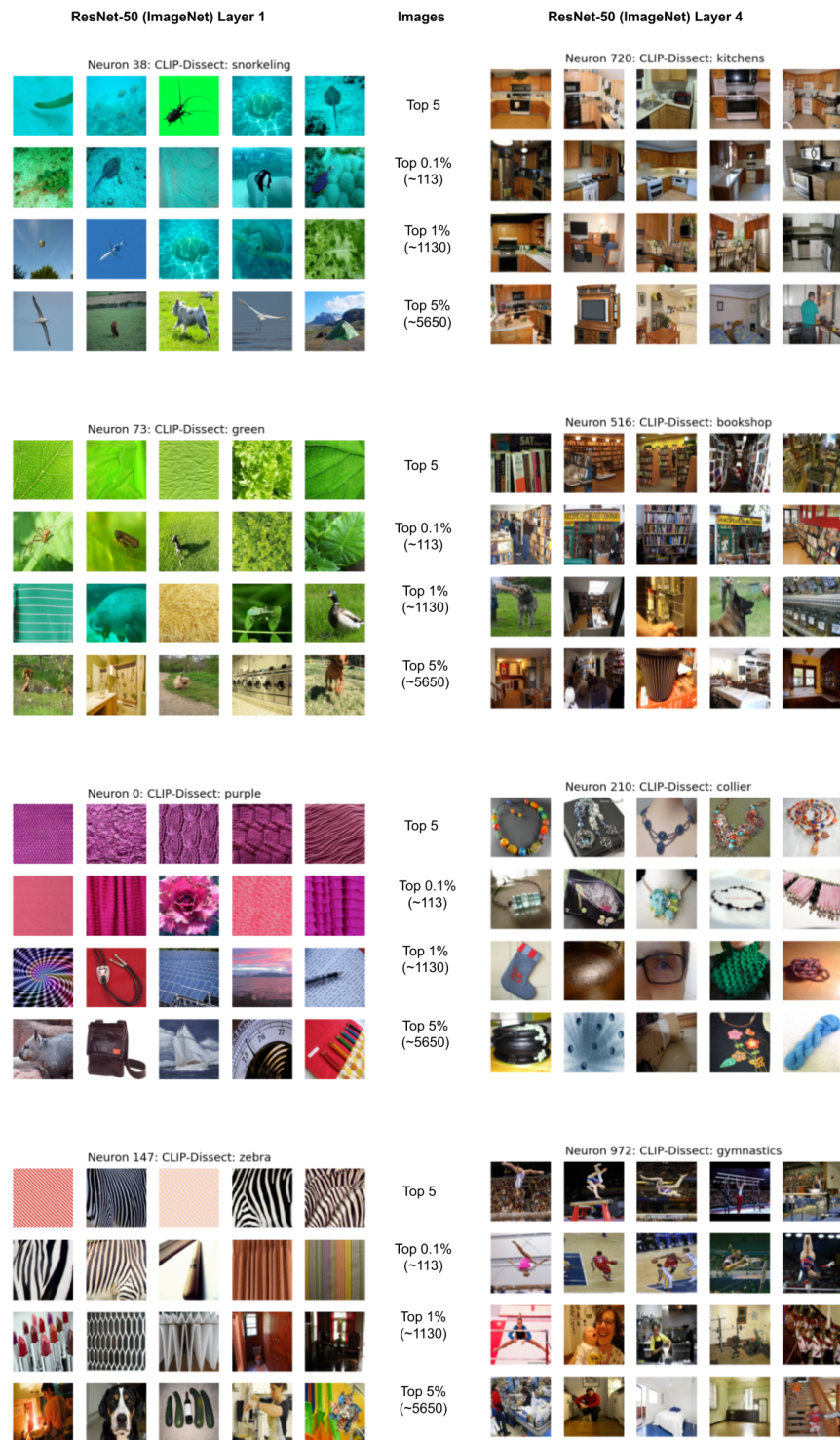


Figure A.8. Most interpretable neurons of ResNet-50, showcasing randomly sampled images of a wide range of most highly activating images for that neuron.

A.1.8 Interpretability cutoff

Our method can also be used to quantify which neurons are 'interpretable'. Since each description for a neuron is associated with a similarity score, and the higher the similarity the more accurate the description, we consider a neuron k with description t as interpretable if $\text{sim}(t, q_k; P) > \tau$. To choose the best cutoff τ , we leverage our experiment in section A.1.10. In particular, we choose the lowest τ such that interpretable neurons will have an average description score of 0.75 or higher (compared to 0.655 of all neurons). This gives us a cutoff threshold $\tau = 0.16$ with the proposed SoftWPMI similarity function. In contrast, the neurons with $\text{SoftWPMI} \leq \tau$ have an average description score of 0.5257, which is lower than that of interpretable neurons (0.75) and all neurons (0.655). Thus, it suggests that the similarity score of SoftWPMI is a useful indicator of description quality. Using this cutoff τ , we find 69.7% of neurons in ResNet-18 (Places-365) and 77.8% of neurons in ResNet-50 (ImageNet) to be interpretable, indicating that around 20-30% of neurons do not have a simple explanation for their functionality, i.e. are 'uninterpretable'.

A.1.9 Qualitative effect of similarity function

Figure A.9 shows the descriptions generated by our CLIP-Dissect when using simple cosine similarity as the similarity function. As we can see the performance is very poor, only adequately describing one of the 8 neurons displayed, highlighting the need for more sophisticated similarity functions we introduced.

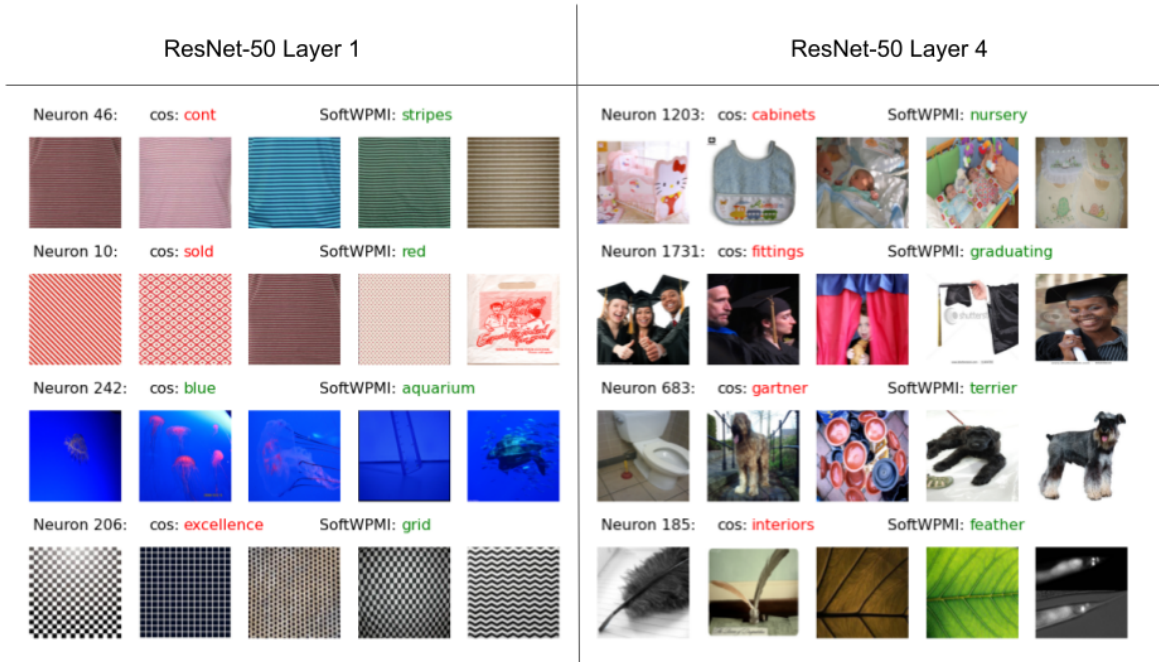
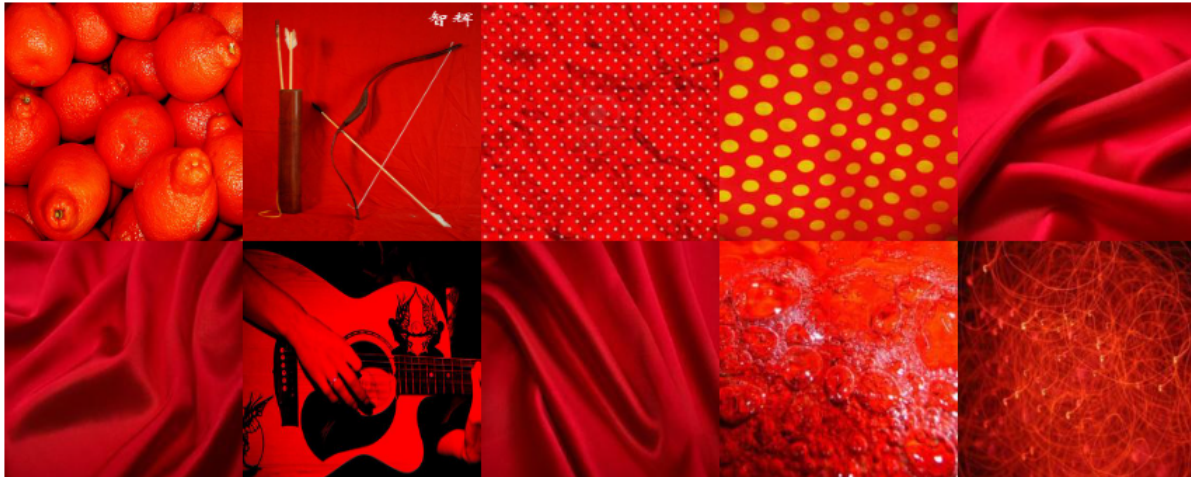


Figure A.9. Explanations for same neurons as Figure 1.1 showcasing the qualitative difference between using a simple cos similarity as the similarity function for CLIP-Dissect vs our best performing SoftWPMI similarity function.

A.1.10 Larger scale experiment on description quality

In this section we perform a larger scale analysis of the neuron description quality provided by CLIP-Dissect. We evaluated the description quality of 50 randomly selected neurons for each of the 5 layers and 2 models studied, for a total of 1000 evaluations. Each evaluator was presented with 10 most highly activating images, and answered the question: "Does the description: '{ }' match this set of images?" An example of the user interface is shown in Figure A.10. Each evaluation had three options which we used with the following guidelines:

- Yes - Most of the 10 images are well described by this description
- Maybe - Around half (i.e. 3-6) of the images are well described, or most images are described relatively well (accurate but too generic, or slightly inaccurate)
- No - Most images are poorly described by this caption



Question 1/50. Description: red

Does the description: 'red' match this set of images?

☐ yes
☐ maybe
☐ no

Submit Reset

Figure A.10. Example of the user interface for our evaluation of description quality.

These evaluations were turned into a numeric score with the following formula: yes:1, maybe:0.5, no:0. Table A.1 shows the average description score across different neurons and evaluators for each of the layers evaluated. This average score can be thought of as the percentage of neurons well described. We observed that overall the descriptions are good for 55-80% of neurons depending on the layer, with the average score across all evaluations being 0.655. In addition we notice that the very early and very late layers are most interpretable, corresponding to clear low or high level concepts, while the middle layers seem to be harder to describe. It is worth noting that we are evaluating random neurons here, i.e. the neurons are selected randomly, so the displayed neuron may not be interpretable in the first place – in many cases when the description does not match are because the neuron itself is not 'interpretable', i.e. there is no simple description that corresponds to the neurons functionality.

Even with the evaluation guidelines described above, these evaluations are subjective,

Table A.1. The average description scores of the CLIP-Dissect descriptions for neurons of different layers. An average score of 1.0 indicates all descriptions match the neurons highly activating images, 0.0 means none do, and 0.5 could mean anything from all neurons 'maybe' match or half the neurons fully match and half don't at all.

Model\Layer	conv1	layer1	layer2	layer3	layer4
ResNet-18 (Places 365)	0.805	0.635	0.635	0.410	0.695
ResNet-50 (ImageNet)	0.815	0.670	0.550	0.640	0.695

and we found that our two evaluators agreed on 68.4% of the neurons with the vast majority of disagreements being between yes/maybe or maybe/no, with only 2.4% of neurons having a yes from one evaluator and no from another. For transparency, we have included all 50 neurons, their descriptions and two evaluator's evaluations (E1, E2) of these descriptions for ResNet-50 layer conv1 in Figures A.11, A.12 and A.13 and for ResNet-50 layer4 in Figures A.14, A.15 and A.16

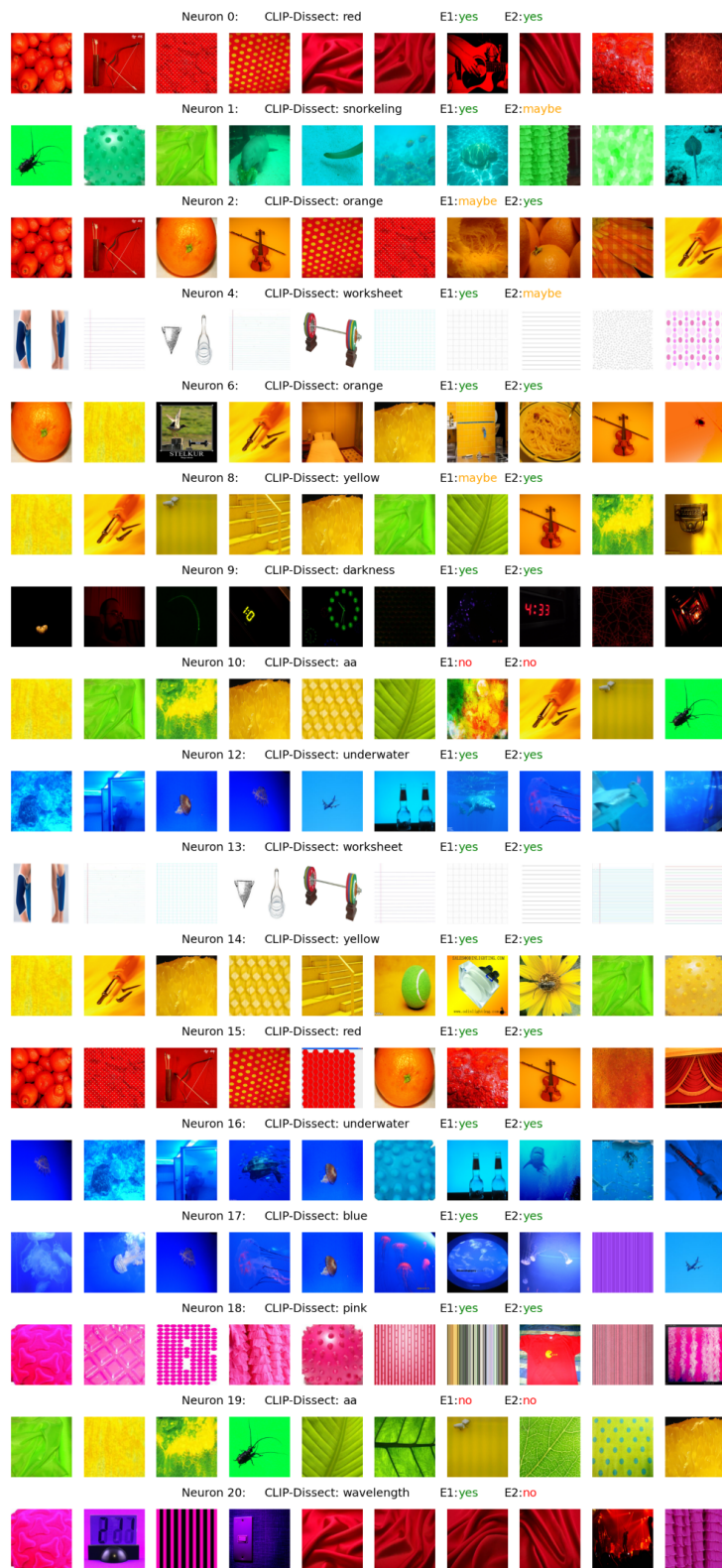


Figure A.11. Random neurons of 'conv1' in ResNet-50 (ImageNet), and E1, E2's evaluation on the description of CLIP-Dissect (PART 1).

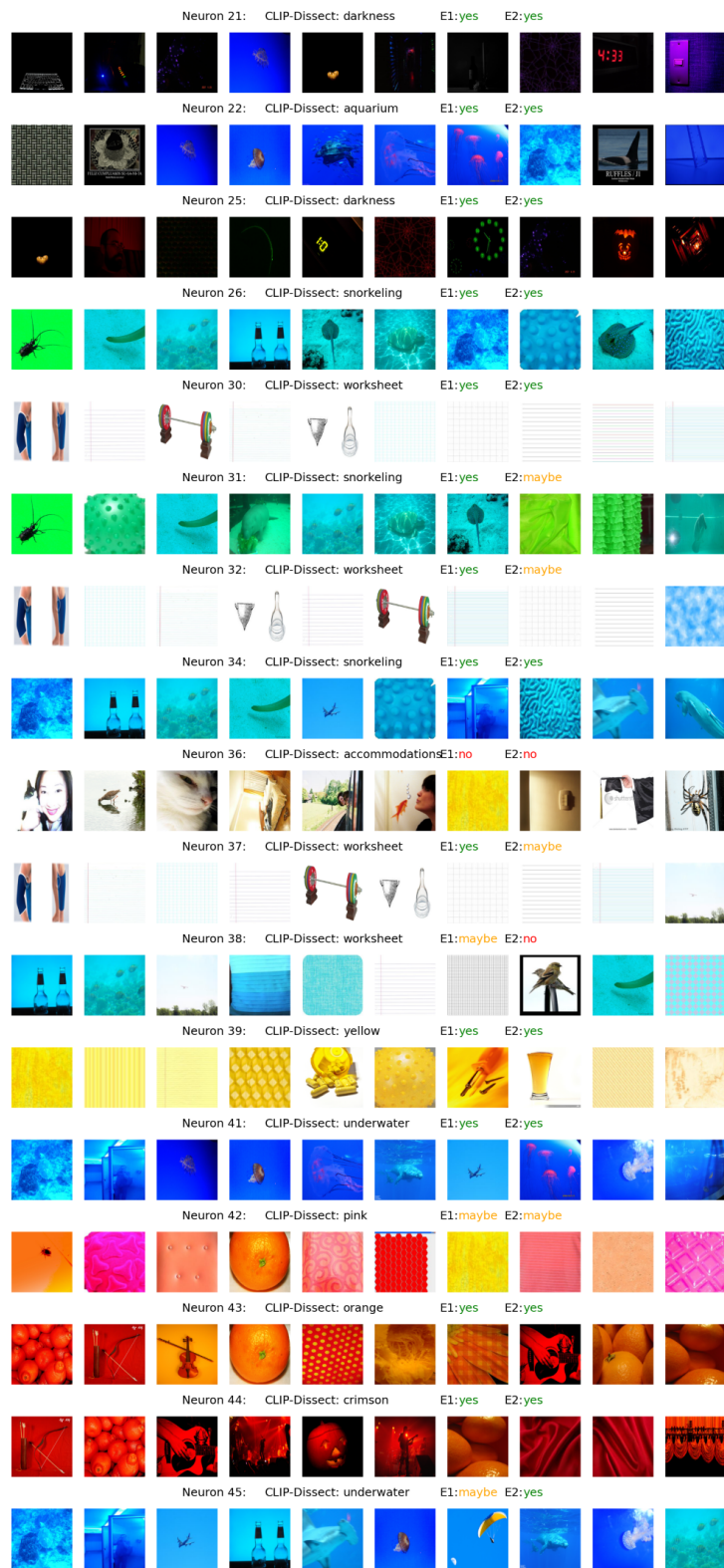


Figure A.12. Random neurons of 'conv1' in ResNet-50 (ImageNet), and E1, E2's evaluation on the description of CLIP-Dissect (PART 2)

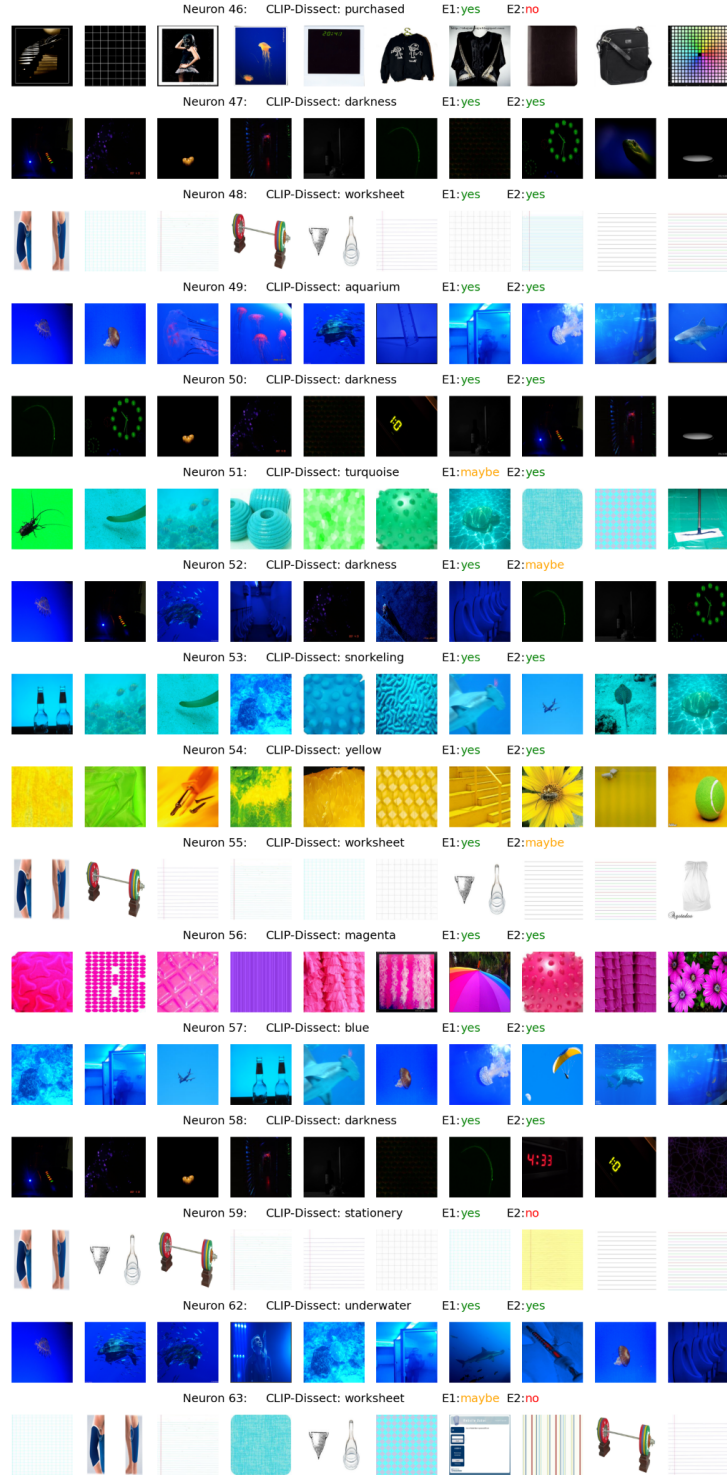


Figure A.13. Random neurons of 'conv1' in ResNet-50 (ImageNet), and E1, E2's evaluation on the description of CLIP-Dissect (PART 3)

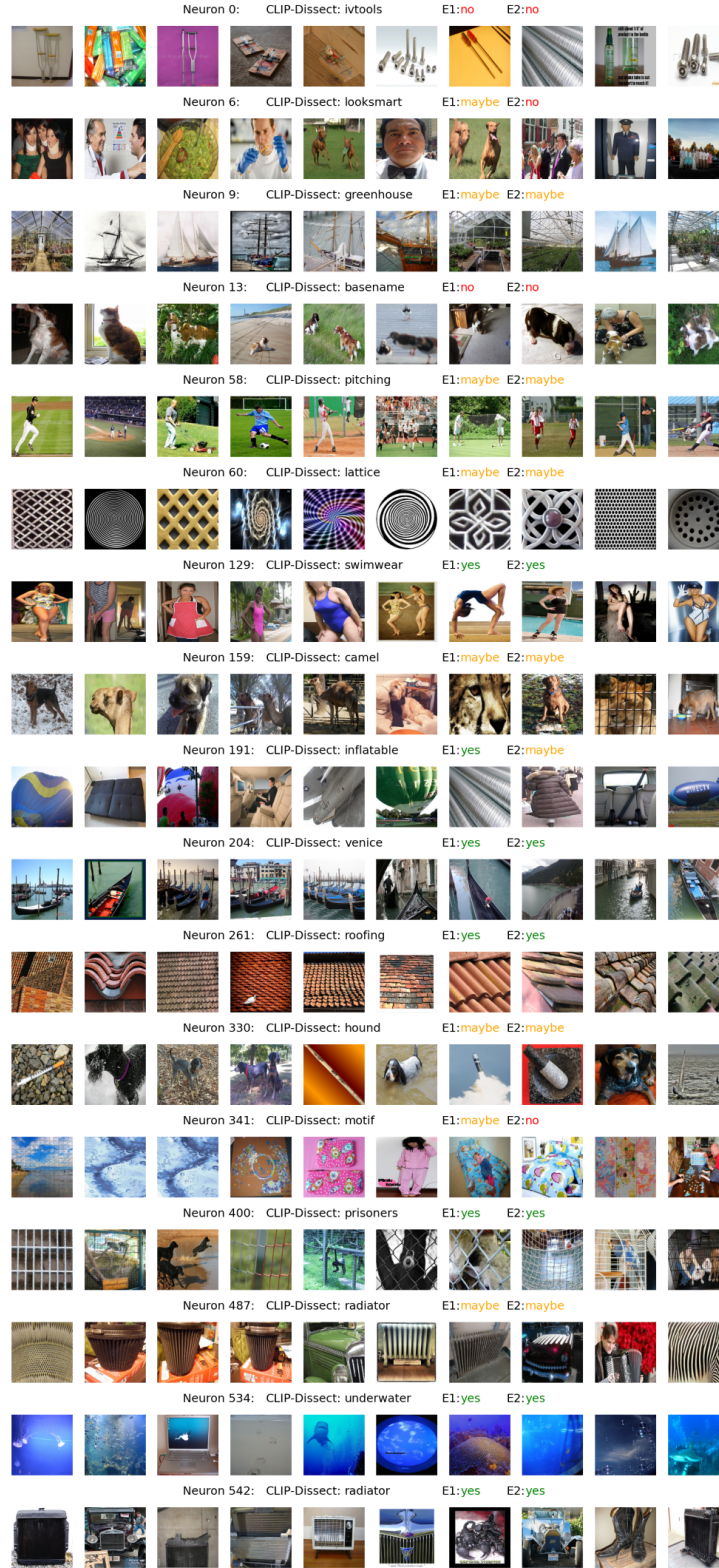


Figure A.14. Random neurons of 'layer 4' in ResNet-18 (Places 365), and E1, E2's evaluation on the description of CLIP-Dissect (PART 1)

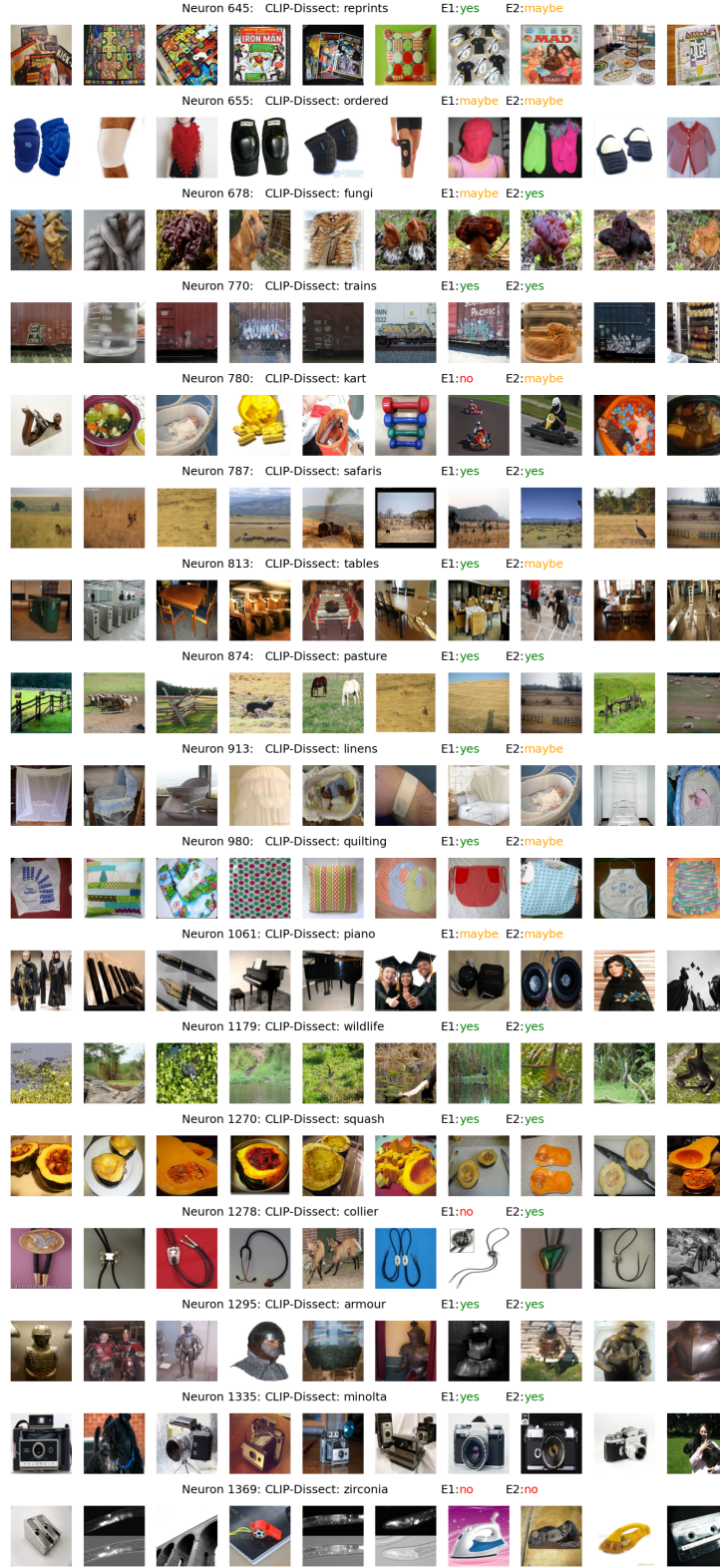


Figure A.15. Random neurons of 'layer 4' in ResNet-18 (Places 365), and E1, E2's evaluation on the description of CLIP-Dissect (PART 2)

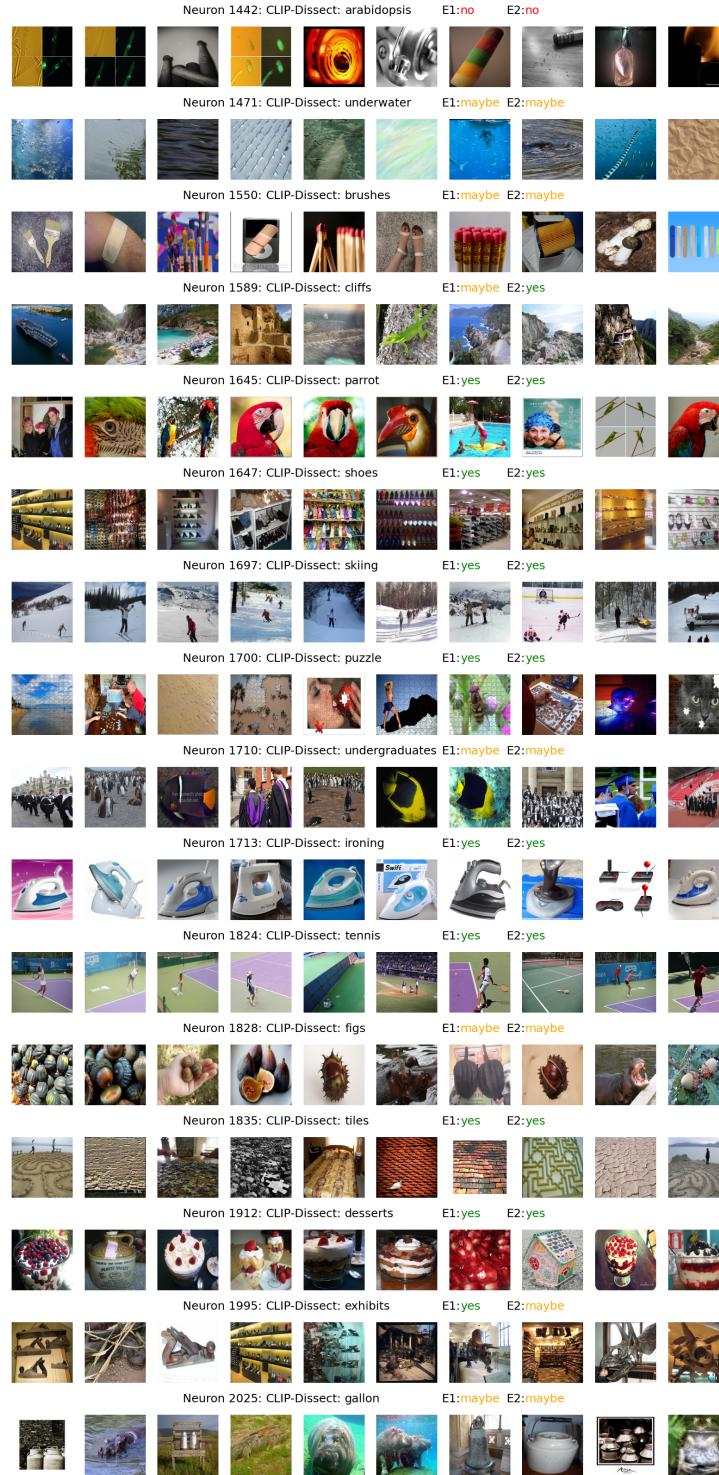


Figure A.16. Random neurons of 'layer 4' in ResNet-18 (Places 365), and E1, E2's evaluation on the description of CLIP-Dissect (PART 3)

A.2 Crowdsourced evaluation of description quality

Note: The results in this section were added after ICLR camera ready to further study the effectiveness of our proposed method, and it is not included in the official ICLR paper.

To get a more objective quantitative measure of the quality of our neuron descriptions, we conducted a large scale study on Amazon Mechanical Turk. The results from this study were in line with our evaluation in Section A.1.10, showing that on average users agree with CLIP-Dissect’s descriptions and that neurons in very early and very late layers are the most explainable.

A.2.1 Setup

We evaluated 200 random neurons per layer (or all neurons for layers with < 200 neurons), in the same 5 layers per model as in Section A.1.10, for both ResNet-50(ImageNet) and ResNet-18(Places). Each neurons’ description was evaluated by 3 raters. The neurons we evaluated were a superset of the neurons evaluated in Section A.1.10. The task interface is similar to that used in Section A.1.10, where we evaluated how well the 10 most highly activating images match the description. However, we changed the rating scale to a five point scale (1 - Strongly Disagree with description, 5 - Strongly Agree with description). The study interface is shown in Figure A.17.

To generate the descriptions we used the union of ImageNet validation dataset and Broden as D_{probe} , the SoftWPMI similarity function, and 20k most common English words as the concept set.

We used raters in the United States, with greater than 98% approval rating and at least 10000 previously approved HITs. Users were paid \$0.05 per task. Our experiment was deemed exempt from IRB approval by the UCSD IRB Board.

Study information

Click to View Study Information

☐ By checking this box I indicate that I am at least 18 years old, have read the study information above, and agree to participate in this research study.

Task

Description for the images: red

Instructions
Shortcuts

Do you agree with the statement below?

Description: "red". Does the description accurately describe most of the above 10 images?

Select an option

Strongly Disagree	1
Disagree	2
Neither Agree nor Disagree	3
Agree	4
Strongly Agree	5

Figure A.17. User Interface for our Mechanical Turk study.

A.2.2 Results

The results of our experiment are shown in Table A.2. We can see that on average, users agreed with our explanations, with the average rating across both models and layers being 3.57 out of 5. In addition, the results agree well with our previous experiment in Section A.1.10, showing earliest and latest layers are most interpretable.

As a baseline/sanity test, we evaluated 10 neurons per layer with their description randomly chosen from the concept set. These descriptions reached an average score of 2.61, which is much lower than CLIP-Dissect explanations (3.57), but still higher than we would expect, indicating there’s some noise in crowdsourced evaluations.

In summary we have shown that our descriptions are clearly better than random on a large scale crowdsourced study, and the findings from our study in Section A.1.10 hold for larger study with more neurons and more evaluators.

Table A.2. Results of our crowdsourced experiment. Each number is the average rating (1-5) across evaluators and neurons for that layer. We can see the average scores are between Neutral and Agree, with little difference between the two models, and highest agreement on the lowest and the highest layers.

Model\layer	conv1	layer1	layer2	layer3	layer4
ResNet-18	3.83	3.58	3.56	3.33	3.71
ResNet-50	3.90	3.48	3.41	3.38	3.50

Appendix B

Supplementary Material for Chapter 2

B.1 Appendix: Additional Information

B.1.1 Limitations

One limitation of our current method is that it does not take into account the location specific activations. This does not change our results on second-to-last layer neurons which is our main focus, but hinders our ability to explain and simulate lower layers well. However this not a fundamental feature of the method, rather an effect of the tools and data we use. Concept-activation matrix P doesn't have to be at the level of images. We can similarly instead run the same method with pixel/superpixel level supervision using for example Broden [6] labels, or an open-vocabulary semantic segmentation model such as [106]. This will however increase the computational cost as we will have to deal with much larger P , and is a limitation we aim to address in the future.

Another limitation is that the simulation scores we achieve, while better than existing methods are still quite low, especially ablation. While this is in line with previous work [9], it is worth discussing. This could be caused by 3 things: 1. Neurons are inherently not interpretable, 2. Our explanations are not good enough or 3. Our simulator is not good enough. While significant gains can definitely be made improving the simulator and explanations (and they can easily be replaced in our pipeline by more powerful models in the future), we believe a large portion of the challenge is caused by the neurons inherently not having a simple explanation. However,

recent work in language models [14, 18] shows promise that we can generate more interpretable units of inspection via methods like sparse autoencoders, to which our method can be applied out-of-the-box.

B.1.2 Additional Related Work: Input Importance

A lot of classical explainable AI methods are focused on a different problem from ours, specifically that of **Input Importance**, where the goal is to answer the question: *Which parts of input x are the most important in the model making prediction $f(x)$?* In contrast, our work is a neuron explanation method, where the goal of our work is to answer the question: *Given a neural network model f , what are the functionalities of each individual neuron?*

For example, the LIME paper [80] is a popular Input Importance work using linear methods for that task. We will use it to illustrate the major difference between the input importance methods and our neuron explanation method.

3 main differences between our method and LIME [80]:

1. **Goal:** LIME aims to explain the final prediction of a model with respect to an input, while we explain a single hidden layer neuron.
2. **Scope:** We produce global explanations, while LIME learns a local explanation that only works close to a specific input.
3. **Input:** Lime learns a linear model in terms of input features (or groups of input features), while our method creates an explanation based on interpretable concepts.

B.1.3 Greedy search algorithm

Algorithm 1 shows the details of our greedy search procedure discussed in Section 2.3. For the tolerance ε , we only add another concept if it improves the correlation (on validation set) between predicted and actual neuron values by more than ε . In our experiments, (v, r, ε) are set to be 10, 10, 0.02 respectively.

Algorithm 1. Greedy search. Python pseudo-code.

```
SC = {} #selected concepts
BC = {} #bad concepts
 $\rho^* = 0$  #best correlation
while  $|SC| < v$  do
    CC = {}
    while  $|CC| < r$  do
        CC.add( $\text{argmax}_{i \notin CC \cup SC \cup BC}(w_{k,i})$ )
    end while
     $\rho' = \rho^*$  #current best corr
    for  $j$  in CC do
        model = train_model( $P_{SC \cup j}^{\text{train}}, q_k^{\text{train}}$ )
         $\rho = \text{get\_correlation}(\text{model}, q_k^{\text{val}})$ 
        if  $\rho < \rho^* + \varepsilon$  then
            BC.add( $j$ ) #concept is bad
        else if  $\rho > \rho'$  then
             $\rho' = \rho$ 
            best_c =  $j$ 
        end if
    end for
    if  $\rho' < \rho^* + \varepsilon$  then
        break #early stop if no improvement
    else
        SC.add(best_c)
         $\rho^* = \rho'$ 
    end if
end while
return SC
```

B.1.4 Explanation length

Table B.1. Average Explanation lengths of our method for second to last layer neurons of different models discussed in Table 2.2.

Target model	LE (Label)	LE (SigLIP)
ResNet-50 (ImageNet)	4.37	4.68
ResNet-18 (Places365)	4.70	4.25
VGG-16 (CIFAR-100)	7.60	6.08
ViT-B/16 (ImageNet)	1.97	1.82
ViT-L/32 (ImageNet)	1.53	1.51

Table B.1 shows the average explanation lengths of our methods for second to last layers of different models. We can see that explanation lengths vary quite a bit, with in particular ViT neurons having short explanations. We think this is likely caused by the fact that the MLP neurons we studied in ViT seem to be more bimodal, either being very uninterpretable or rather monosemantic and interpretable, both of which result in short explanations in our case. If more similar explanation lengths are desired, this can be achieved by tuning the tolerance parameter of the greedy search for each model.

B.2 Appendix: Ablations

B.2.1 Different simulator model.

Since both our simulator and SigLIP explainer model use a similar (but different) model from the SigLIP family, it is natural to ask whether the good performance is caused by this similarity of models. To address this, we redo our simulation with correlation scoring experiment (Table 2.2) using a simulator model from original CLIP [77] family, namely CLIP-ViT-L-14-336. Results are shown in Table B.2. We can see that all scores are lower than with our original simulator model (SigLIP-SO400M-14-386), likely because this CLIP model is not as powerful and it is not designed for multilabel classification required from a simulator. Our Linear Explanation (SigLIP)’s performance is reduced the most by switching the simulator model, indicating that using a related simulator model does boost its performance, but it is still the best method with a different simulator.

Table B.2. Correlation scores of different explanations in layer4 of the models, using CLIP ViT-L-14-336 as the simulator model. Results are similar to Table 2.2, but all methods score lower, probably due to weaker simulator model.

Target model	Net Dissect	MILAN	CLIP-Dissect	LE (Label)	LE (SigLIP)
ResNet-50 (ImageNet)	0.1003	0.0789	0.1725	0.2545	0.2961
ResNet-18 (Places365)	0.1626	0.1279	0.1951	0.2825	0.3154

B.2.2 Different concept sets

In table B.3, we tested how the choice of concept set affects the results of our LE(SigLIP). We compared our original (class labels + Broden labels + common nouns) against two alternatives: 20k, a set of 20,000 most common English words from [68], as well as only using the ImageNet class labels. We can see our concept set noticeably outperforms both alternative choices. Interestingly it even outperforms the larger 20k concept set, perhaps because it is more noisy and lacking some more precise terms(class names) useful in describing ImageNet images.

Table B.3. Average simulation correlation scores for final layer neurons of ResNet-50(ImageNet) layer4 of LE(SigLIP) with different choices of concept set. We can see our concept set performs the best. Number in brackets represents the size of the concept set

Concept set	Correlation score
Original (8438)	0.3772
20k (20,000)	0.3561
ImageNet labels (1000)	0.3378

B.2.3 Different explainer model

We also conducted a study testing the importance of the choice of explainer model for the performance of LE(SigLIP). In particular, we replaced our original explainer SigLIP-ViT-L-16-384 with CLIP ViT-L-14-336 from the original CLIP paper [77]. From the results in Table B.4 we can see a stronger explainer model makes a big difference, but even when using a weaker CLIP model our method outperforms other methods.

Table B.4. Comparison of different explainer models for LE(SigLIP) on laeyr4 neurons of ResNet-50(ImageNet).

Explainer model	Correlation score
Original (SigLIP-ViT-L-16-384)	0.3772
CLIP ViT-L-14-336	0.3243

B.2.4 No greedy search

To assess how important our greedy search procedure described in section 2.3.2 is for explanation quality, we evaluated our method without the greedy search procedure by simply taking the top-k concepts of our relatively sparse w_k (with weights retrained with top-k concepts only) as the final explanation for the neuron. Simulation results are shown in Table B.5. Overall the results are mixed, showing we can reach roughly similar quality explanations without the greedy search, but we still find using it preferable as greedy search can dynamically determine the desired explanation length for each neuron, giving longer descriptions to more complex neurons and simple explanations to monosemantic neurons.

Table B.5. Average correlation score of different explanations for neurons in layer4 of ResNet-50(ImageNet). Number in brackets is the average description length.

Method	LE(label)	LE(siglip)
Original	0.2924 (4.37)	0.3772 (4.68)
No greedy search (top-5)	0.3136 (5.00)	0.3767 (5.00)
No greedy search (top-4)	0.2962 (4.00)	0.3548 (4.00)

B.2.5 Scaling for Simulation with Ablation Scoring

In section 2.4 we define our ablation scoring function $\alpha_{init}(k, \mathcal{D}, E, c, d)$. This requires finding the optimal scaling parameters c^* and d^* . For our main results we used the optimization method for finding these parameters:

Optim:

$$c^*, d^* = \arg \max_{c, d} \alpha_{init}(k, \mathcal{D}_{probe}^{val}, E, c, d) \quad (\text{B.1})$$

In contrast, [9] use a closed form solution intended to maximize the explained variance to select these parameters. In their method (which we will call **Norm**):

Norm:

Let

$$\rho_{val}(k, E) = \sum_{x \in \mathcal{D}_{probe}^{val}} \frac{\hat{s}(x, E) \cdot \hat{g}(A_k(x))}{|\mathcal{D}_{probe}^{val}|} \quad (\text{B.2})$$

where $\hat{s}$ and $\hat{g}(A_k(x))$ are normalized to have mean 0 and standard deviation 1 on the validation set. Then:

$$c^* = \rho_{val}(k, E) \cdot \frac{\sigma(\mathcal{D}_{probe}^{val}, g(A_k(\cdot)))}{\sigma(\mathcal{D}_{probe}^{val}, s(\cdot, E))} \quad (\text{B.3})$$

$$d^* = -c^* \cdot \mu(\mathcal{D}_{probe}^{val}, s(\cdot, E)) + \mu(\mathcal{D}_{probe}^{val}, g(A_k(\cdot))) \quad (\text{B.4})$$

where:

$$\mu(\mathcal{D}, f) = \frac{1}{|\mathcal{D}|} \sum_{x_i \in \mathcal{D}} f(x_i) \quad (\text{B.5})$$

$$\sigma(\mathcal{D}, f) = \sqrt{\frac{\sum_{x_i \in \mathcal{D}} (f(x_i) - \mu(\mathcal{D}, f))^2}{|\mathcal{D}|}} \quad (\text{B.6})$$

This is theoretically justified as maximizing the explained variance of the neuron. However, our goal is not to maximize explained variance, which leads us to believe direct optimization might be more effective (**Optim**). In table B.6 we compare the two methods when evaluating explanations for layer4 neurons of ResNet-50(ImageNet). We can see the optimization method results in noticeably (20-50%) larger ablation scores overall, but doesn’t affect the order between explanation methods.

However, optim requires several (100 in our experiments) backwards passes over the validation data, and as such is much more computationally costly. For second to last layer neurons this can be done very efficiently with only a few seconds per neuron, but for evaluating earlier layers the **Norm** strategy is likely better.

Table B.6. Testing the scaling function for ablation scoring with simulation. Average ablation score across all neurons in layer4 of ResNet-50(ImageNet). We can see optimizing the scaling parameters c, d results in a 20-50% increase in ablation scores, while not affecting the ordering between methods.

Scaling Method	Network Dissection	MILAN	CLIP-Dissect	LE(Label)	LE(SigLIP)
Norm	0.0118 ± 0.0004	0.0092 ± 0.0004	0.0187 ± 0.0005	0.0334 ± 0.0007	0.0602 ± 0.0009
Optim	0.0165 ± 0.0003	0.0137 ± 0.0003	0.0215 ± 0.0004	0.0433 ± 0.0007	0.0727 ± 0.0008

B.3 Appendix: Additional Results

B.3.1 Additional layers.

ResNet lower layer neurons.

Table B.7 and B.8 show the correlation scores of different methods across different layers of ResNet-50(ImageNet) and ResNet-18(Places365). We can see performance significantly degrades on the lower layers, which we think is partially caused by neuron functions being harder to describe, but also because our simulator cannot simulate pixel level activations, which becomes more impactful when describing lower layer neurons. However we can still see Linear Explanation, especially with SigLIP continues to outperform all existing methods. Finally in Table B.9 we report the results on two different layers on VGG-16.

Table B.7. Average correlation scores of explanations by different methods on different layers of ResNet-50 trained on ImageNet. We can see correlation scores noticeably decrease on lower layers, but Linear Explanations perform the best on all layers. The number in brackets represents average number of concepts per explanation.

Target layer	Network Dissection	MILAN	CLIP-Dissect	Linear Explanation (Label)	Linear Explanation (SigLIP)
layer1	0.0313	0.0430	0.0670	0.1008 (1.98)	0.2313 (3.32)
layer2	0.0324	0.0242	0.0527	0.1037 (1.91)	0.2180 (3.22)
layer3	0.0794	0.0773	0.1045	0.1308 (2.32)	0.2652 (3.69)
layer4	0.1231	0.0920	0.1871	0.2924 (4.37)	0.3772 (4.68)

Table B.8. Average correlation scores of explanations by different methods on different layers of ResNet-18 trained on Places365. The number in brackets represents average number of concepts per explanation.

Target layer	Network Dissection	MILAN	CLIP-Dissect	Linear Explanation (Label)	Linear Explanation (SigLIP)
layer1	0.0223	0.0344	0.0603	0.1384 (2.42)	0.2574 (3.66)
layer2	0.0261	0.0154	0.0419	0.1351 (2.41)	0.2803 (3.84)
layer3	0.0727	0.0768	0.0704	0.1593 (2.73)	0.2842 (3.71)
layer4	0.2038	0.1557	0.2208	0.3388 (4.70)	0.4372 (4.25)

Table B.9. Average correlation score across neurons in different layers of VGG16(CIFAR100). We tested the second to last layer (features[4]) which is a fully connected layer, as well as the last convolutional layer features[43]. Number in brackets is average explanation length. Both layers reach quite high correlation scores, but require long explanations, indicating the neurons are polysemantic.

Model	Target layer	CLIP-Dissect	LE(label)	LE(SigLIP)
VGG16 (CIFAR100)	classifier[4] (second to last, fc)	0.2298	0.4330 (7.60)	0.4970 (6.08)
VGG16 (CIFAR100)	features[43] (last CNN)	0.2616	0.5107 (7.58)	0.5430 (6.07)

Different layers of ViT

In table B.10, we study the interpretability of different layers in ViT models. We can see that the residual stream neurons are in general noticeably less interpretable than the MLP neurons. This is likely caused by the fact that the residual stream does not (mostly) have a "privileged basis", which means individual neurons are not more interpretable than random directions in the activation space. In other layers/architectures such a privileged basis is created by axis-aligned activation functions such as ReLU. See [21] for more discussion on privileged bases. Because of this, we focus on analyzing the MLP neurons in our work. This is in line with work investigating individual neurons in transformer language models, which typically focus on these MLP neurons. Interestingly, we found that some of the neurons in the MLP layers were extremely interpretable with the highest correlation scores > 0.9 , which we did not see in CNN models. On the other hand, the MLP layer also had several lowly activating/dead neurons that were not particularly interpretable, bringing down the average.

When evaluating the final layer neurons of ViT (both residual stream and MLP), we only recorded the activations of the CLS token, as this is the only part passed on to the classification head, and the other activations do not matter. For earlier ViT layers, we took average across the CLS token and all spatial activations, though further research is needed to better know how important the CLS token is compared to spatial activations at different layers of the network.

Table B.10. Average simulation correlation scores of different methods for different layers of ViT-B-16. We can see MLP layers are more interpretable on average than the residual stream.

Model	Target layer	CLIP-Dissect	LE(label)	LE(SigLIP)
ViT-B/16 (ImageNet)	11/11 - Residual Stream (end)	0.0326	0.0813	0.1455
ViT-B/16 (ImageNet)	11/11 - MLP	0.1722	0.3243	0.3489
ViT-B/16 (ImageNet)	10/11 - Residual Stream (end)	0.0860	0.1293	0.2643
ViT-B/16 (ImageNet)	10/11 - MLP	0.1067	0.1940	0.3088

B.3.2 Dead neurons in ViT

When investigating the neurons in Vision Transformer, we came across a curious phenomenon: several neurons of the MLP layers are completely dead, i.e. don’t activate on any inputs. Such neurons are not meaningful to explain, and seem to waste model capacity. This was especially noticeable in our Experiments with ViT-L/16, where every single neuron in the last transformer block is dead. We tested this further, and found that the last (23rd) transformer layer does not do anything, and can be deleted from the model without affecting classification accuracy at all. Because of this weirdness, we did not include ViT-L/16 in our main results and instead used ViT-L/32.

In Table B.11 we quantify the number of dead neurons in the second to last layer of different models (same layers as in Table 2.2). We defined a neuron as dead if it’s maximum activation was < 0.01 across the probing data. We can see other ViT models suffer from some dead neurons but not very many, with ViT-L/16 being an outlier. The CNN based models have no dead neurons.

Table B.11. Fraction of dead neurons in different models.

Target model	ResNet-50 (ImageNet)	ResNet-18 (Places-365)	VGG-16 (CIFAR-100)	ViT-B/16 (ImageNet)	ViT-L/32 (ImageNet)	ViT-L/16 (ImageNet)
Dead neurons:	0%	0%	0%	5.27%	6.57%	100%

In Table B.12, we report the average correlation scores on ”active” neurons only, i.e. we don’t count the dead neurons for models with some dead neurons. Overall this does not make a

big change as the fraction of dead neurons was quite small, but most scores improve by around 5% over Table 2.2, which is roughly the fraction of dead neurons.

Table B.12. Average correlations scores of different explanation methods without dead neurons on ViT. We can see this improves average correlation scores by around 5% over Table 2.2

Target model	CLIP-Dissect	LE (Label)	LE (SigLIP)
ViT-B/16 (ImageNet)	0.1807 +- 0.004	0.3399 +- 0.005	0.3629 +- 0.005
ViT-L/32 (ImageNet)	0.0590 +- 0.002	0.1984 +- 0.004	0.2254 +- 0.004

B.3.3 Additional Qualitative Results

In Figures B.1, B.2, B.3, B.4, B.5 we display our explanations as well as previous work for several neurons from layer4 of ResNet-50, and show that our explanations generally capture the neurons behavior well, even if it doesn't have a simple function. In addition, we notice that the LE(Label) and LE(SigLIP) methods typically produce similar concepts, but SigLIP version often returns higher weights for these. This is likely due to differences in concept activation matrices P used by the two models, and highlights the need to calibrate the magnitude of predictions when simulating to align these with the simulator.

In Figures B.6, B.7, B.8, B.9, B.10 we display similar figures for other models. Some interesting things we can see are that in the case on extremely monosemantic neurons, such as ViT-B/16 neuron 1541 in Figure B.9, our method correctly returns only one concept explanation for that neuron, and that concept is more accurate than that of CLIP-Dissect. We also found a rather interesting neuron activating on both military ships/aircraft as well as bald eagles as shown in Figure B.10, which our methods described very accurately.

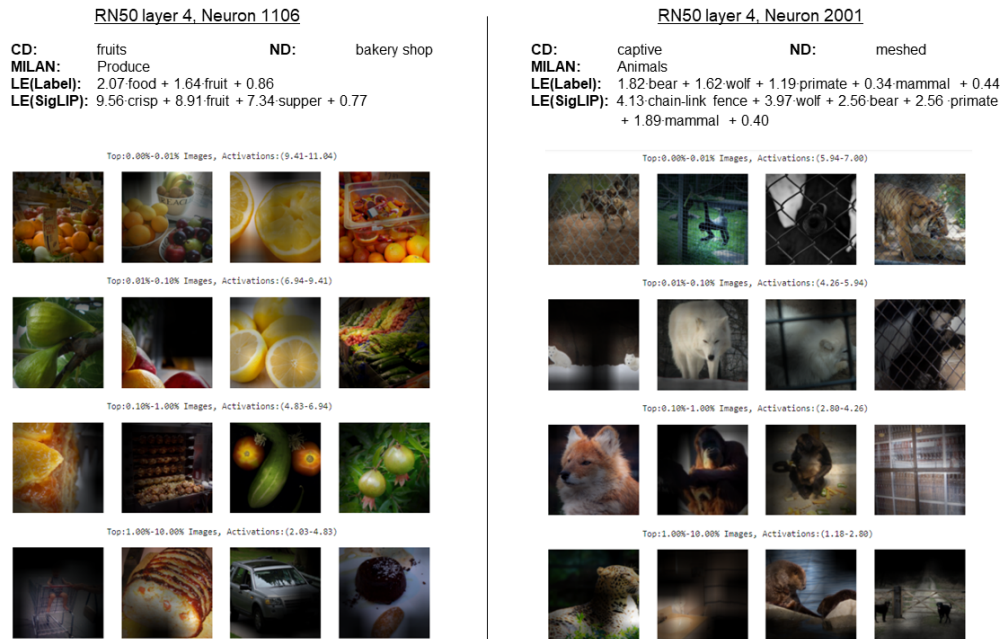


Figure B.1. Example interpretable neurons.

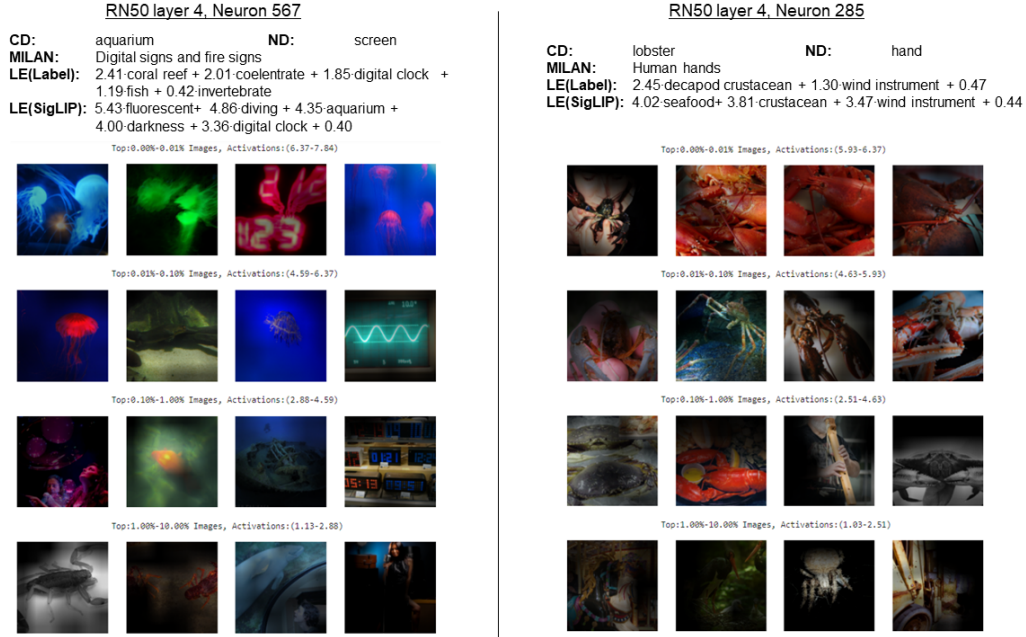


Figure B.2. Example interpretable neurons.

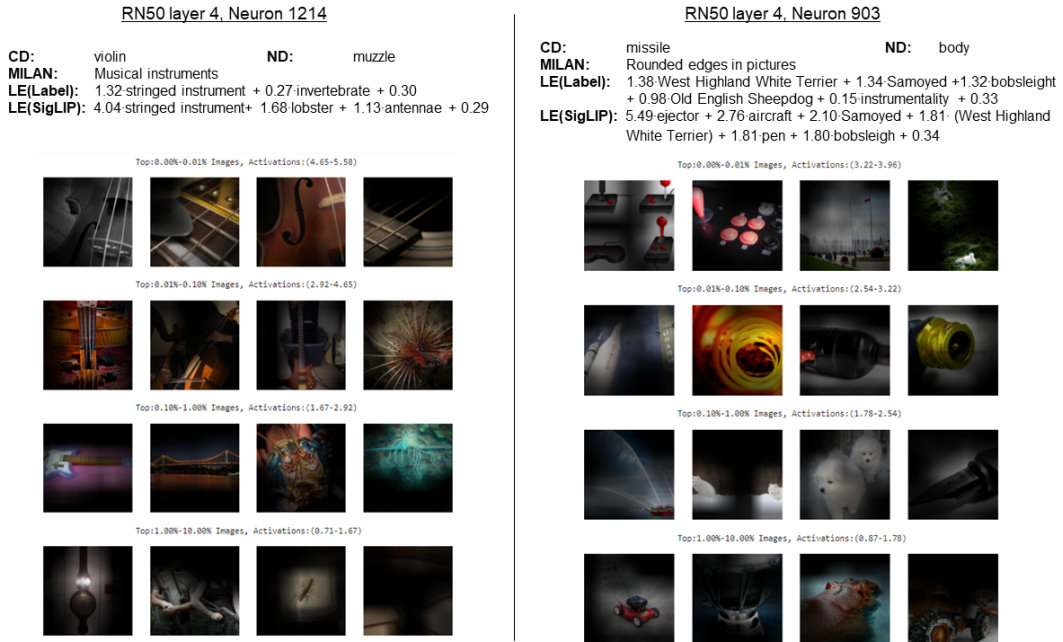


Figure B.3. Randomly selected neurons.

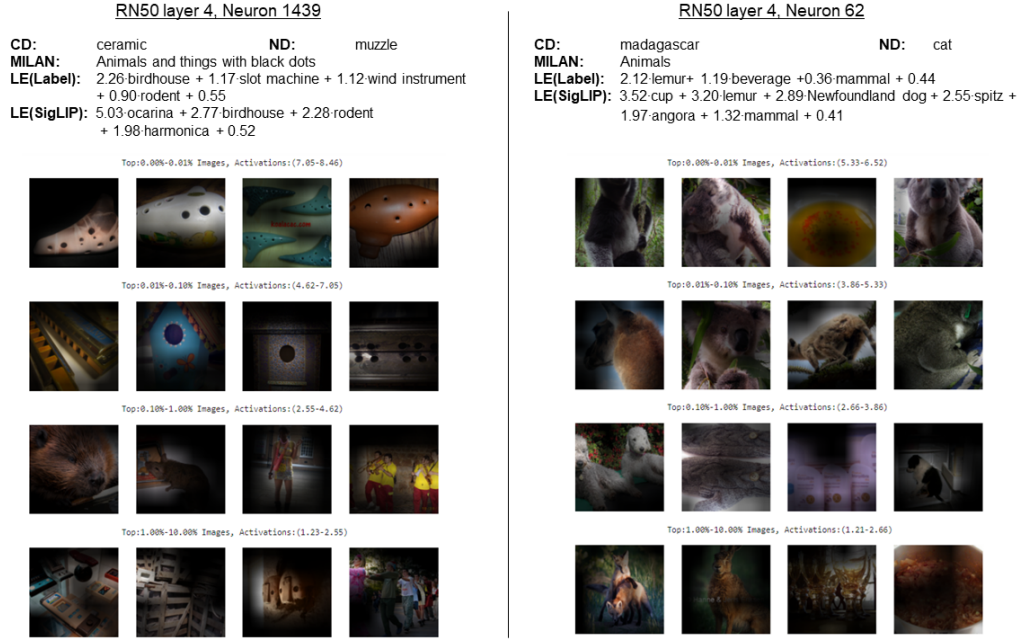


Figure B.4. Randomly selected neurons.

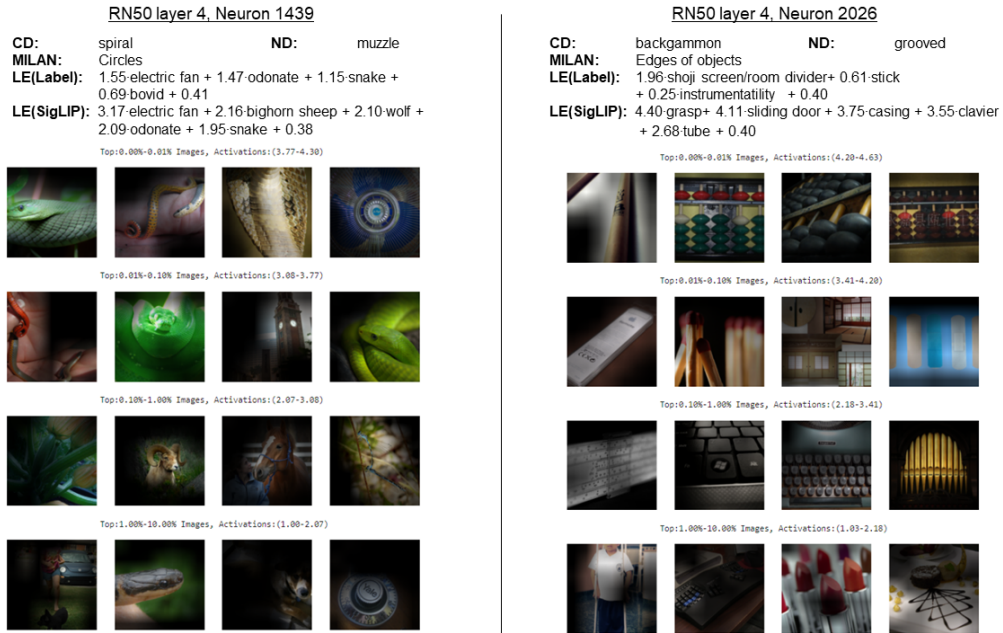


Figure B.5. Randomly selected neurons.

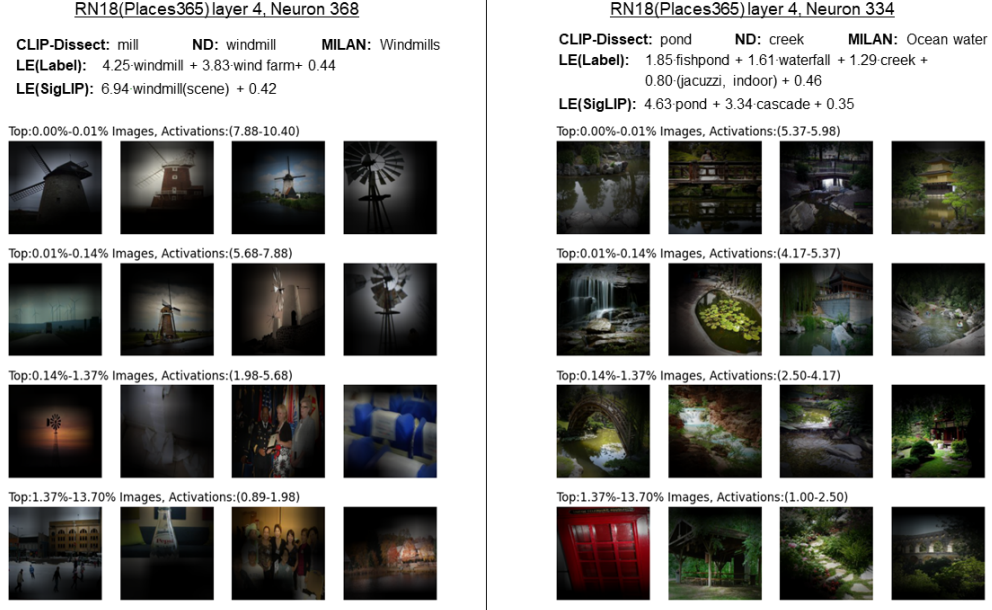


Figure B.6. Example interpretable neurons of ResNet-18(Places365).

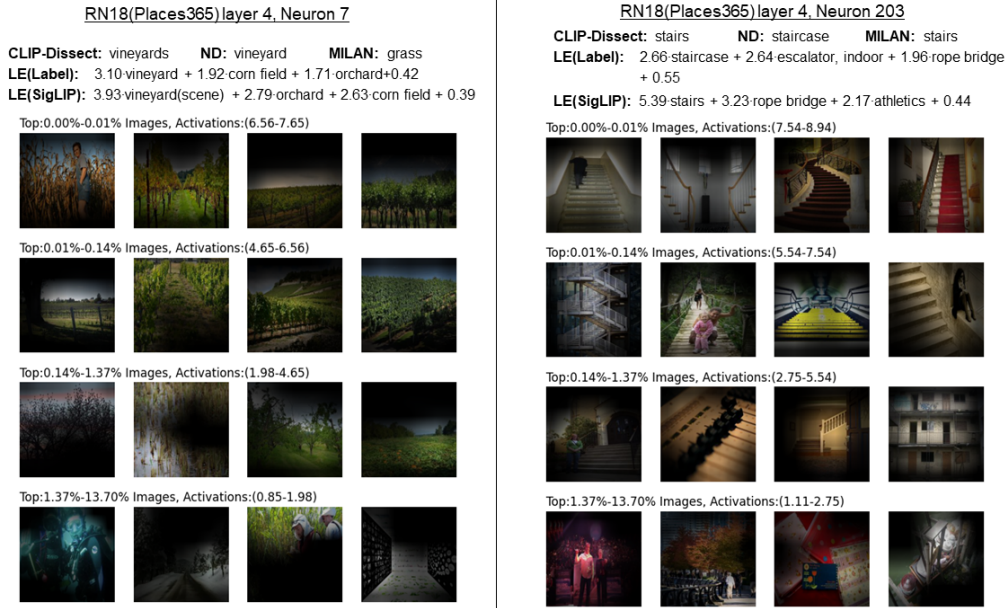


Figure B.7. Example interpretable neurons of ResNet-18(Places365).

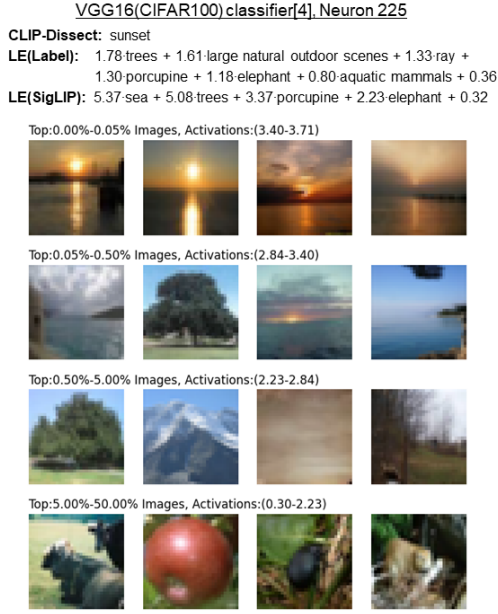


Figure B.8. Example interpretable neurons of VGG-16(CIFAR-100).

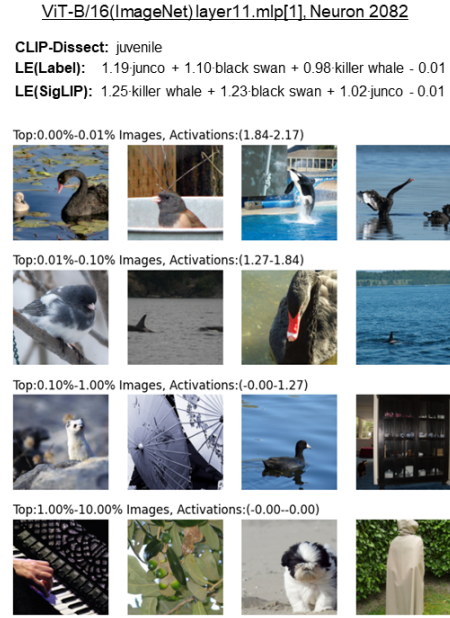
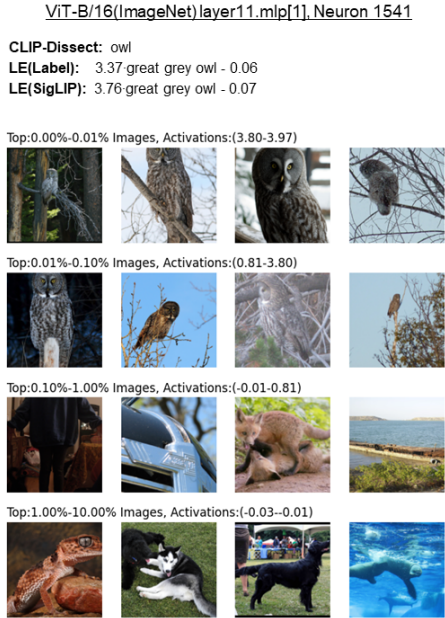


Figure B.9. Example interpretable neurons of ViT-B/16(ImageNet).

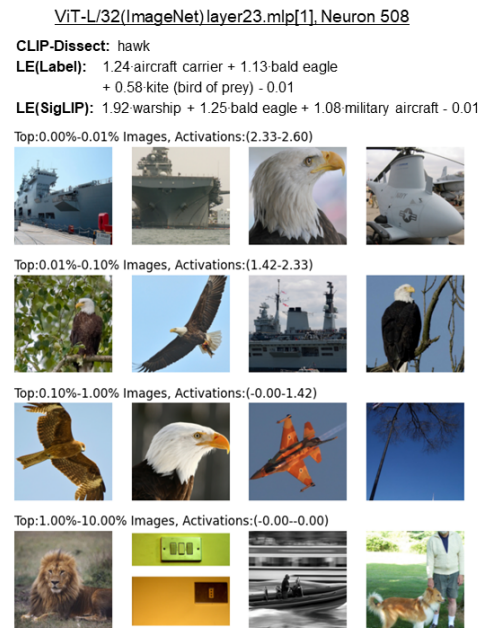


Figure B.10. Example interpretable neurons of ViT-L/32.

Appendix C

Supplementary Material for Chapter 3

C.1 Metric Definition & Details

C.1.1 Binarization

Many popular metrics in the literature require inputs to be binary such as [6, 42]. Since neuron activations and concept activations from some sources are continuous, we need to binarize these vectors. We denote this with the binarization function $B : \mathbb{R}^n \rightarrow \{0, 1\}^n$.

Typically for neuron activation a_k , following previous work we use $B = \text{top}_\alpha$, where we take top α fraction of activations to be 1, and others to be 0. We formalize this as $\text{top}_\alpha(z)$:

$$[\text{top}_\alpha(z)]_i = \begin{cases} 1 & \text{if } z_i \geq b_\alpha; \\ 0 & \text{otherwise} \end{cases} \quad (\text{C.1})$$

where b_α satisfies $\sum_{i=1}^n \frac{\mathbb{1}[z_i \geq b_\alpha]}{n} = \alpha$, and $z \in \mathbb{R}^n$. For example, if $\alpha = 0.05$, then top_α has 1's for inputs with activations in top-5%, and 0 for others. Note α is a hyperparameter needed for all binary similarity functions. We typically select α independently for each metric by finding the value that performs the best on a small validation split of neurons. For concept vectors c_t , we

usually binarize simply by rounding, denoted as $B = r$, where:

$$[r(z)]_i = \begin{cases} 1 & \text{if } z_i \geq 0.5 \\ 0 & \text{if } z_i < 0.5 \end{cases} \quad (\text{C.2})$$

C.1.2 Simulation vs Classification Framing

Simulation:

In our definitions in section 3.3 we use the neuron activation a_k as the *ground truth*, and concept presence c_t as our prediction. This corresponds to framing the evaluation as *simulation*, i.e. trying to predict neuron activation based on concept value. Given this, we can express the Binary Classification Confusion Matrix elements (True Positive (TP), False Positive (FP), False Negative (FN) and True Negative (TN)) in terms of the vectors a_k and c_t as:

$$\begin{aligned} \text{TP} &= B(a_k) \cdot B(c_t), \text{FP} = \overline{B(a_k)} \cdot B(c_t), \\ \text{FN} &= B(a_k) \cdot \overline{B(c_t)}, \text{TN} = \overline{B(a_k)} \cdot \overline{B(c_t)}, \end{aligned}$$

where $\overline{B(\cdot)}$ represents element-wise NOT operation on the binary vector (equivalent to $\mathbf{1} - B(\cdot)$ where $\mathbf{1}$ is a vector of all 1's.)

This corresponds to seeing the explanation from a **simulation** point of view, i.e. our goal is to predict how the neuron activates, based on the neuron's explanation and current inputs. This gives us binary classification metric definitions that are aligned with those of [42].

Classification:

However this is an arbitrary choice, and we could just as well define c_t as the *ground truth* and a_k as the prediction. This corresponds to a classification view, where our goal is to use neuron k as a classifier for concept t . In terms of metrics, this doesn't change the definitions for True Positive (TP) or True Negative (TN), but it switches the places of False Positive and False Negative, i.e.

$$\text{TP}_{cls} = B(a_k) \cdot B(c_t) = \text{TP}_{sim}$$

$$\text{FP}_{cls} = B(a_k) \cdot \overline{B(c_t)} = \text{FN}_{sim}$$

$$\text{FN}_{cls} = \overline{B(a_k)} \cdot B(c_t) = \text{FP}_{sim}$$

$$\text{TN}_{cls} = \overline{B(a_k)} \cdot \overline{B(c_t)} = \text{TN}_{sim}$$

This change in framing also affects for metrics which are not symmetric in terms of False Positives and False Negatives. For example, $\text{Recall}(\text{simulation}) = \text{Precision}(\text{classification})$, and $\text{Precision}(\text{simulation}) = \text{Recall}(\text{classification})$.

$$\text{Recall}(\text{simulation}) = \frac{\text{TP}_{sim}}{\text{TP}_{sim} + \text{FN}_{sim}} = \frac{\text{TP}_{cls}}{\text{TP}_{cls} + \text{FP}_{cls}} = \text{Precision}(\text{classification})$$

The other binary metric that is sensitive to this framing is balanced accuracy. The metric we call Balanced Accuracy in Appendix C.1.3 corresponds to the simulation version of Balanced Accuracy, so for completeness sake we also include Inverse Balanced Accuracy (Appendix C.1.3), which is the classification version.

Of the non-binary metrics, AUC and AUPRC are sensitive to this framing, and the standard versions AUC and AUPRC we define in Appendix C.1.3 are $\text{AUC}(\text{simulation})$ and $\text{AUPRC}(\text{simulation})$, while we also include classification versions of these metrics labeled as Inverse AUC/AUPRC.

We note that some related works use the simulation framing, while others such as [114] use the classification framing, which leads to conflicting definitions of Precision and other metrics.

C.1.3 Additional Metric Definitions

1. **Accuracy:** Standard binary accuracy. Percentage of time the neuron activation matches concept activation.

$$s_{\text{Acc}} = \frac{TP + TN}{TP + FP + FN + TN} = \frac{B(a_k) \cdot B(c_t) + (\mathbf{1} - B(a_k)) \cdot (\mathbf{1} - B(c_t))}{n} \quad (\text{C.3})$$

2. **F1-score:** F1-score is the harmonic mean of precision and recall, and can be expressed as:

$$s_{F1} = \frac{2 \cdot B(a_k) \cdot B(c_t)}{\|B(a_k)\|_1 + \|B(c_t)\|_1} \quad (\text{C.4})$$

3. **Balanced Accuracy:** A version of accuracy designed for imbalanced datasets that averages the accuracy on positive and negative inputs.

$$s_{BA} = \frac{B(a_k) \cdot B(c_t)}{2\|B(a_k)\|_1} + \frac{(\mathbf{1} - B(a_k)) \cdot (\mathbf{1} - B(c_t))}{2\|(\mathbf{1} - B(a_k))\|_1} \quad (\text{C.5})$$

4. **Inverse Balanced Accuracy:** Balanced accuracy but we consider a_k to be the prediction and c_t to be the ground truth.

$$s_{IBA} = \frac{B(a_k) \cdot B(c_t)}{2\|B(c_t)\|_1} + \frac{(\mathbf{1} - B(a_k)) \cdot (\mathbf{1} - B(c_t))}{2\|(\mathbf{1} - B(c_t))\|_1} \quad (\text{C.6})$$

5. **AUC:** Area under ROC curve also known as AUROC, can be efficiently calculated as:

$$s_{\text{AUC}} = \frac{\sum_i |B(a_k)_i=0| \sum_j |B(a_k)_j=1| \mathbb{1}[c_{ti} < c_{tj}] + 0.5 \cdot \mathbb{1}[c_{ti} = c_{tj}]}{\|B(a_k)\|_1 \|1 - B(a_k)\|_1} \quad (\text{C.7})$$

6. **Inverse AUC:** Area under receiving-operating-characteristics(AUROC) curve, where we consider a_k to be the prediction and c_t to be the ground truth(classification framing).

$$s_{IAUC} = \frac{\sum_i |B(c_t)_{i=0}| \sum_j |B(c_t)_{j=1}| \mathbb{1}[a_{ki} < a_{kj}] + 0.5 \cdot \mathbb{1}[a_{ki} = a_{kj}]}{\|B(c_t)\|_1 \|1 - B(c_t)\|_1} \quad (\text{C.8})$$

7. **Spearman Correlation:** The Spearman correlation is equivalent to the Pearson Correlation between the ranks of elements.

$$s_{Spear} = \frac{1}{n} \frac{(R(a_k) - \mu(R(a_k))) \cdot (R(c_t) - \mu(R(c_t)))}{\sigma(R(a_k)) \sigma(R(c_t))} \quad (\text{C.9})$$

Here R is a function that elementwise returns the ranks of each element.

8. **Cosine similarity:** The standard cosine similarity between two vectors.

$$s_{cos} = \frac{a_k \cdot c_t}{\|a_k\|_2 \|c_t\|_2} \quad (\text{C.10})$$

9. **WPMI:** Weighted pointwise-mutual information. A version of this objective is used by [39] and [68] to generate explanations, and by [68] to evaluate said explanations.

$$s_{WPMI} = \sum_{i|B(a_k)_i=1} [\log(c_{ti}) - \lambda \log(\mu(c_t))] \quad (\text{C.11})$$

In the above equations n is the length of a_k and c_t , μ calculates the mean of the vector and σ its standard deviation. λ is a hyperparameter and R is the rank operator, which transforms each element to its rank, with smallest element becoming 1 and largest n .

10. **AUPRC:** Area Under Precision-Recall Curve(AUPRC) is a popular metric for measuring classification performance, in particular for imbalanced data. While we are not aware of a closed form solution, it can be calculated as:

- (a) Calculate precision and recall at each threshold τ_i , where threshold contains distinct values of c_t . Recall $R_i = \frac{B(a_k) \mathbf{1}(c_t \geq \tau_i)}{\|B(a_k)\|_1}$, precision $P_i = \frac{B(a_k) \mathbf{1}(c_t \geq \tau_i)}{\|\mathbf{1}(c_t \geq \tau_i)\|_1}$.

(b) Calculate area under precision-recall curve using numerical integral:

$$s_{AUPRC} = \sum_n (R_i - R_{i-1}) P_i$$

AUPRC outputs values in $[0, 1]$ range.

11. **Inverse AUPRC:** Same as AUPRC, but with a different framing so we flip c_t and a_k in the calculations.

See Tables C.1 and C.2 for additional details on our metrics.

C.1.4 Equivalences

During our analysis we also notice that certain separate metrics are equivalent or very similar to each other.

Correlation and Cosine similarity:

Calculating correlation between two vectors is equal to normalizing each vector to have mean 0 and then taking their cosine similarity, as shown below:

Let $\hat{x} = x - \mu(x)$ for any vector x . Then

$$\begin{aligned} \text{Cosine}(\hat{a}_k, \hat{c}_t) &= \frac{\hat{a}_k \cdot \hat{c}_t}{\|\hat{a}_k\|_2 \|\hat{c}_t\|_2} = \frac{(a_k - \mu(a_k)) \cdot (c_t - \mu(c_t))}{\sqrt{n}\sigma(a_k)\sqrt{n}\sigma(c_t)} = \\ &= \frac{1}{n} \frac{(a_k - \mu(a_k)) \cdot (c_t - \mu(c_t))}{\sigma(a_k)\sigma(c_t)} = \text{Correlation}(a_k, c_t) \end{aligned} \quad (\text{C.12})$$

This explains why the two perform very similarly in our evaluations.

IoU and F1-score:

Below we show that IoU and F1-score are very closely related. In fact, F1-score can be written as a monotonously increasing function of IoU. This means that for any vectors x_1, y_1, x_2, y_2 , $\text{IoU}(x_1, y_1) < \text{IoU}(x_2, y_2) \rightarrow \text{F1}(x_1, y_1) < \text{F1}(x_2, y_2)$, so for the purposes of comparing similarities

they behave identically, and the choice of which one to use doesn't matter. As their performance was exactly the same in all tasks, we report them in the same row in Table 3.4.

Intersection over Union (IoU) also known as Jaccard index is defined as

$$\text{IoU} = \frac{TP}{TP + FP + FN} \quad (\text{C.13})$$

while F1-score also known as Dice-score is defined as:

$$\text{F1} = \frac{2TP}{2TP + FP + FN} \quad (\text{C.14})$$

Now

$$\text{F1} = \frac{2TP}{2TP + FP + FN} = \frac{2TP}{TP + FP + FN} \cdot \frac{TP + FP + FN}{2TP + FP + FN} \quad (\text{C.15})$$

$$= 2\text{IoU} \cdot \left(\frac{2TP + FP + FN}{TP + FP + FN} \right)^{-1} = \frac{2 \cdot \text{IoU}}{\text{IoU} + 1} \quad (\text{C.16})$$

Which is monotonously increasing for $0 \leq \text{IoU} \leq 1$. So using either metric gives the same comparative results.

Table C.1. Definition of commonly-used binary classification metrics. Here, TP, FP, TN, FN refer to true positive, false positive, true negative and false negative, respectively.

Metric	Definition	Range
Recall	$TP / (TP + FN)$	$[0, 1]$
Precision	$TP / (TP + FP)$	$[0, 1]$
F1-score	$2TP / (2TP + FP + FN)$	$[0, 1]$
IoU	$TP / (TP + FP + FN)$	$[0, 1]$
Accuracy	$(TP + TN) / (TP + FP + TN + FN)$	$[0, 1]$
Balanced accuracy	$[TP / (TP + FN) + TN / (TN + FP)] / 2$	$[0, 1]$
Inverse balanced accuracy (classification version of balanced accuracy (see App. C.1.2))	$[TP / (TP + FP) + TN / (TN + FN)]$	$[0, 1]$

Table C.2. Definition of other commonly-used metrics. Here, $x, y \in \mathbb{R}^N$ are two real vectors, $\bar{x}, \bar{y}$ refer to the mean of x, y .

Metric	Definition	Range
AUC (swap x and y to get inverse AUC)	$\frac{\sum_{y_i=1} \sum_{y_j=0} [\mathbf{1}\{x_i > x_j\} + 0.5 * \mathbf{1}\{x_i = x_j\}]}{ y = 1 y = 0 }$	$[0, 1]$
Correlation	$\frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$	$[-1, 1]$
Spearman correlation	Replace x, y to corresponding rank $R(x), R(y)$ in correlation	$[-1, 1]$
Cosine	$\frac{\sum x_i y_i}{\ x\ _2 \ y\ _2}$	$[-1, 1]$
WPMI	$\log p(x y) - \lambda \log(p(x))$	$(-\infty, \infty)$
MAD	$\frac{\sum_{y_i=1} x_i}{ y_i = 1 } - \frac{\sum_{y_i=0} x_i}{ y_i = 0 }$	$(-\infty, \infty)$

C.2 Discussion

C.2.1 Generative Model based Evaluation and missing labels test

c_t from Generative Models: Evaluation methods that use generative models to generate new data [51, 86, 88] effectively define the concept vector $[c_t]_i = 1$ if x_i was generated with prompt t , and 0 otherwise. However, this concept vector is incomplete, as some of "negative" examples that were not generated with that prompt may still contain the concept. In effect, the concept vector c_t is actually naturally missing labels similar to c_t^- from our missing labels test. This is because the generated inputs serve as positive labels for c_t , but the negative inputs are often taken to just be all inputs from a dataset, even though some of actually do have the concept t . So the c_t we get from generative models is actually randomly missing a potentially large portion of the labels. Because of this, most evaluation methods using generative c_t in Table 3.1 use methods that fail the missing labels test such as Inverse AUC and MAD. In fact, this is desirable with this concept source, as the missing labels in c_t do not affect the explanation score with these metrics. However, these metrics alone should not be relied for evaluation.

Potential Solutions and Future Directions: The best way to use generative models in evaluation might be to combine them with another evaluation that doesn't fail the missing labels test, as is done by [42] who measure Precision using generated data, and measure Recall on existing data with model based pseudo-labels and combine these results into F1-score. A useful evaluation in this vein could also be using a crowdsourced evaluation to measure Recall, and combining that with generative model based evaluation of Precision. Another option would be to not use the c_t derived from prompt directly, but instead use another model to estimate c_t on all inputs, not just generated ones which would avoid the missing labels problem. Finally an interesting direction would be to look into combinations of other metrics, for example could AUC and Inverse AUC be combined similarly to how Recall and Precision combine into F1-score. We explore these combination metrics further in Appendix C.5.6.

C.2.2 Limitations

Framework Limitations:

Not every evaluation of neuron descriptions can fit into our framework. Below we split these into few separate cases and discuss whether each case represents a limitation of the framework or not:

- **Evaluating Multiple Inputs at once:** Our evaluation framework assumes that the presence of a concept is estimated separately for each input. Many human study based evaluations (e.g. [6], [68]) instead evaluate a group of inputs at once, asking questions like "How well does *concept* match this group of images?". However we believe this is simply a less precise/less objective way of asking whether the concept matches each input separately and does not in general represent a significant limitation for the framework.
- **Comparing similarity to "correct" explanation:** Another approach to evaluate neuron descriptions is to compare how close they are to a "correct" description, typically in a text-embedding space. For example, this is the main evaluation used by MILAN [39], where they generate "correct" explanation by asking Mechanical Turk workers to describe neurons based on their most highly activating inputs. We do not think this a very reliable way to evaluate explanations, because it relies on the assumption that there exists a single "correct" text-based explanation for each neuron (and that we have some way of finding it), and we do not think this is the case for many real neurons because of issues like polysemanticity and non-verbal concepts like specific graphical patterns. For these hard-to-interpret neurons it is better to just measure how well our explanation matches the neuron like the metrics in our framework do.
- **Non-text based explanations:** While we focus on text based concepts t in our paper, the framework works on non-textual concepts just as well, as long as we have some way of generating a concept vector c_t for that concept. For example, the evaluation of [117] uses

a group of highly activating inputs as the concept, and then asks workers whether a new input is similar to those inputs or not. Despite this difference, it can be described neatly within our framework.

- **Output Based Neuron Explanations:**

For clarity, it is useful to divide the neural network $f(x)$ into two parts, $f^{0:l}$ and $f^{l+1:L}$, where 0 corresponds to the input layer, $f^{i:j}$ represents the i through j 'th layers of the neural network $f(x)$, l is the layer of the neuron we are interested in and L is the total number of layers in the network. Then:

$$f(x) = f^{l+1:L}(f_k^{0:l}(x), f_{-k}^{0:l}(x)), \quad (\text{C.17})$$

where $f_k^{0:l}(x)$ is the activation of neuron of interest k in layer l , while $f_{-k}^{0:l}(x)$ is the activations of all the other neurons in layer l .

Alternatively to **Input-based** neuron explanations we define in section 3.2.3, a neuron explanation could instead be **Output-based**, that is an interpretable approximation of the following function:

$$\text{-- Unit Activation} \rightarrow \text{Output: } z \rightarrow f^{l+1:L}(z, f_{-k}^{0:l}(x))$$

Where z is a real number (e.g. intervened value). This function describes how changes in the unit activation change the final network output, as has been the focus of a few recent papers on neuron explanations [28, 36].

As we can see from the above notations, output and input-based explanations are different problems, and may require different methods to solve and evaluate. In our paper, we focus on evaluating explanations of **Input-based**, i.e. the function $x \rightarrow f_k^{0:l}(x)$, as this is more common in existing evaluations and can be applied more generally, for example to explain linear combinations of neurons that don't have a direct effect on the output such as

TCAV [48] vectors. While evaluating Output-based explanations is also important, this may require different methods and is outside the scope of our current work.

This may be the most significant limitations of our framework, as we believe measuring Output based explanations is equally important, and an ideal neuron explanation should describe both functions (potentially with different explanations). However, improving the evaluation of Input-based explanations is already a significant contribution towards better evaluation practices. In addition, while currently our framework is meant for Input based evaluation only, we believe many of the ideas and metrics we discussed could be useful in evaluating output based explanations. For example, in a generative models we could use the same metrics to measure similarity between unit activation and the presence of a specific concept in the output. However in output-based evaluations there are additional considerations such as measuring difference in outputs when changing the unit activation that may be more important. We believe extending this framework or creating a similar one for output-based evaluations is an important direction for future work.

Experimental Result Limitations:

Overall we are quite confident in the generality of our results on the missing/extra labels test (Table 3.3) as they are consistent across final/hidden layer neurons and different datasets, and we showed theoretically that they are caused by poor metric performance on imbalanced data. Importantly, this theoretical result is independent of the data domain, type of concepts or the type of unit in question.

However, it is good to note that passing these sanity checks does not guarantee that the metric is a good metric, but failing them does indicate a metric should not be relied on. This is similar to sanity checks proposed by [2], which have been quite influential in the field of saliency maps/input importance estimation.

On the other hand, our comparison results in Table 3.4 are mostly focused on final layer neurons or other units where we have ground truth available such a concept neurons inside a

CBM. While they consistently prefer certain metrics, it is possible that these final layer neurons are systematically different from other units we are interested in such as hidden layer neurons, and these results should not be relied on too strongly.

Labeling Cost:

One of the main reasons existing crowd-sourced evaluations only measure *Recall* is it’s lower evaluation cost, as we only need to only evaluate c_t on highly activating inputs, i.e. where $B(a_k) = 1$, because the values of c_t on other inputs do not affect the score. In contrast, evaluating most other metrics requires knowledge of the full c_t . Similar cost issues are also faced when using an expensive LLM based simulation pipeline to predict c_t as done by [9], causing evaluation metrics to often be evaluated on a small subset instead.

Crowd-sourced evaluation of the entire c_t will likely give noisy results as most inputs do not contain the concept. To avoid this, another approach is to oversample highly activating inputs similarly to Top-and-random sampling, though more effective sampling strategies likely exist. This can be done without failing the extra labels test if the proper sampling correction is applied to correct for the bias introduced by the non-uniform sampling. As our paper is focused on the correctness of different metrics, not the ease of measuring them, and the effective sampling/user study design for non-recall metrics is a rather large and complex topic that does not fit within the scope of the current paper, but it is a promising direction for future work.

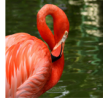
An alternative way to reduce labeling costs is to use a combination of metrics, for example evaluating F1-score by combining a crowd-sourced evaluation of Recall with a generative model based evaluation of Precision.

C.2.3 Motivating Example for Sanity Checks

Here, we illustrate the motivating example discussed in Section 3.4.1 in the following Figure. Figure C.1 displays an example neuron that only activates on images containing pets to highlight how similarity scores are calculated. We can see that only measuring Recall gives a perfect score to explanations that are too generic (*animal*), while only measuring precision

favors explanations that are too specific(*dog, cat*).

Example neuron activating on pets only

Probing data D_{test}						
$B(a_k)$	1	1	1	0	0	0
c_{dog}	1	0	1	0	0	0
c_{cat}	0	1	0	0	0	0
c_{pet}	1	1	1	0	0	0
c_{animal}	1	1	1	1	1	1

Results of different evaluation metrics:

Concept t	s_{Recall}	$s_{\text{Precision}}$	s_{IoU}
Dog	$2/3 = 0.67$	$2/2 = 1$	$2/3 = 0.67$
Cat	$1/3 = 0.33$	$1/1 = 1$	$1/3 = 0.33$
Pet	$3/3 = 1$	$3/3 = 1$	$3/3 = 1$
Animal	$3/3 = 1$	$3/6 = 0.5$	$3/6 = 0.5$

- Correct concept **Pet** gets perfect score with all metrics
- Recall **incorrectly** gives perfect score to superclass *Animal*
- Precision **incorrectly** gives perfect score to subclasses *Dog* and *Cat*

Figure C.1. A hypothetical neuron that only activates on pets (dogs or cats). When comparing different evaluation metrics, we can see recall cannot distinguish between the correct concept (Pet) and a concept that is too generic (Animal), while precision favors concepts that are too specific (Dog, Cat). IoU can unambiguously determine the correct concept.

C.2.4 Additional Related Works

Evaluation of individual neuron explanations.

Table C.3 shows an expanded version of our comparison table (Table 3.1 in the main text section 3.4) of existing evaluation methods. It can be seen in Table C.3 that prior work [6, 68, 67, 4, 42, 37, 65, 54, 50, 117, 16, 51, 9, 69, 14, 96, 67, 68, 86, 88] only use 1-2 metrics for evaluation and typically did not discuss or justify why the metric should be used. Among all the prior work in Table C.3, we believe that the most similar work to ours is [42], which has focused on evaluating individual neuron explanations in language models. In particular, they discover a discrepancy between the evaluation metrics, i.e. neurons with very high Correlation(top-and-random) score can still have relatively low F1-scores. However, different from our work their scope is much more specific and they do not provide analysis comparing different evaluation metrics or justification on why they use F1-score specifically. Their findings are in line with ours, as we found that Correlation(top-and-random) fails the extra labels test, while F1-score passes our sanity checks.

Overall we find many evaluations in previous works to be lacking, either due to using poor metrics that fail our sanity checks, or using very small sample sizes. In addition, some popular methods like TCAV [48] completely lack evaluation of whether the concept directions they learn are good. To our best knowledge, no human-study has been conducted using metrics that pass our sanity checks, instead most existing human-studies only measure Recall or a similar metric, and running such a study would be valuable for better understanding of unit interpretability and/or explanation methods.

Known Concept Evaluation: Similar to our experiments in Section 3.5, many previous works such as [68, 85, 63, 86] have utilized neurons or units with known concepts to evaluate explanation quality. However, these papers typically focus on evaluating explanation methods or individual explanations, while our focus is on *meta-evaluation* to understand which metrics are best to use for evaluation, meaning our methods and motivations for the evaluation are noticeably

different from these previous works.

Interpretability Illusions:

[10] discovered an interpretability illusion for BERT-models, which shows that neurons that look interpretable when only looking at their most highly activating inputs on one probing dataset $\mathcal{D}$, seem to be doing something completely different when analyzed on another dataset. This highlights the importance of good choice of probing dataset, or confirming explanations on multiple datasets. However, we believe there is another reason for these illusions, namely that the authors only look at highly activating inputs (equivalent to Recall in our framework), which is not a reliable metric. Overall, our goal is to reduce such interpretability illusions, i.e. the appearance of interpretability when something is really not interpretable, by encouraging more rigorous evaluation.

Sanity Checks.

The sanity checks proposed in this paper (Missing and Extra Labels test) are inspired by the sanity checks [2] proposed for saliency maps, which has had a large impact in guiding that field towards more faithful explanations. However the topic is very different, since [2] focus on local input-importance instead of global neuron explanations in our paper. Similarly, the specific tests proposed by [2] are also very different from ours.

Concept extraction.

Extracting interpretable concepts from a learned representation is a common challenge and relevant for finding individual units to evaluate in our framework. This can either be supervised as proposed by [48] where the concepts are specified by human and labels are provided. This approach is also used by linear probing based work such as [37]. Later, a series of works[30, 113, 26] was proposed to automatically extract concepts from model activations, without human supervision, i.e. unsupervised. [24] claimed that concept extraction could be regarded as a dictionary learning problem. Recently Sparse Autoencoders [18, 96] have also gained popularity as an unsupervised concept extraction method. Note that most unsupervised

concept extraction methods are discovering "units" defined in our work that are not directly understandable to humans, and require an explanation method to provide human-understandable explanations. Our work focuses on the evaluation of the explanations of those concepts.

Human-centered evaluation on concept-based models.

Human-centered evaluation of model explanation[49, 13] has drawn attention from the XAI community. Recently, [47] collected human evaluation of XAI explanations as a benchmark for explanation methods. These works provide important techniques and inspiration for evaluating explainability, but almost all existing work is focus on local feature importance explanations, which is very different from our work on global neuron-level explanations.

Evaluation Metrics for Input-Importance Explanations

The field of input-importance explanations has seen an evolution in the evaluation metrics used, with initial focus on finding the features that humans think are important. Later metrics such as deletion and insertion proposed by [75] allow for more principled evaluation of the explanation fidelity, i.e. whether it actually matches what the model in vision models, and [25] extends the deletion metric to natural language settings. [41] proposes the Remove And Retrain (ROAR) framework as an alternative method for evaluating the quality of input-importance explanations by evaluating whether a model retrained on data without the most important pixels can still solve the task. [8] propose additional checks, such as measuring whether an explanation method has high sensitivity, with the intuition that similar inputs should have similar explanations. While this line of work focuses on a different topic (local input importance explanations) instead of global neuron-level explanations, and mostly proposes new metrics without significant meta-evaluation, it highlights the importance of good evaluation metrics in explainable AI.

C.3 Theoretical Missing/Extra Labels Test and Concept Imbalance

In this section we analyze the effects of missing/extra labels test on a simpler toy setting, where neuron k 's activations a_k perfectly match the concept labels of concept t_k , i.e. $a_k = c_{t_k}$. In this setting, we assume binary neuron and concept activations, i.e. the neuron's activation is 1 if concept is present on the input, and 0 if its not. Consequently, we do not need to perform additional binarization of concept activations with top α like we did in previous sections.

In this simplified setting, our missing and extra labels tests correspond to being able to differentiate between three concepts as defined in section 3.4:

1. c_{t_k} : The perfect predictor for neuron a_k , with precision=1 and recall=1.
2. $c_{t_k}^-$: (Missing labels) This concept has Precision=1 since whenever a concept is present, the neuron is also active, and Recall of 0.5 since only half to inputs where the neuron activate now have the concept.
3. $c_{t_k}^+$: (Extra labels) Inverse from above, this concept has Precision=0.5 and Recall=1.

We then measure the average score difference Δs across neurons:

- Missing labels test: $\Delta s(k) = \mathbb{E}_{c_{t_k}^-} [s_M(a_k, c_{t_k}^-) - s_M(a_k, c_{t_k})]$
- Extra labels test: $\Delta s(k) = \mathbb{E}_{c_{t_k}^+} [s_M(a_k, c_{t_k}^+) - s_M(a_k, c_{t_k})]$

A good metric should be able to reliably differentiate between these concepts. Interestingly we find that the ability of most metrics to differentiate greatly depends on whether the data is balanced or not.

Since the neuron activations perfectly align with concept t and are binary, the only parameter that can effect the results of our missing and extra labels test is the activation frequency of concept t , i.e. what fraction of inputs $x \in \mathcal{D}$ contain concept t . Following the notation in

section C.4, we denote this fraction as γ . Note technically it should be $\gamma + \eta$, but $\eta = 0$ in this case with perfect match between concept and neuron. We then test whether a metric passes the test on different values of γ , using simulated data on tables C.6 and C.7. Each number is the average result from 1000 evaluations with 500,000 datapoints each. In addition, in Section C.4, we derived a closed form solution to the binary metrics under missing or extra labels as a function of γ and other parameters. This simplifies nicely when we consider an ideal neuron with $a_k = c_t$, and we can derive the expected result of missing/extra labels test as a simple function of γ alone in Table C.9. These theoretical results perfectly agree with our simulated results.

C.3.1 Results

In Tables C.4 and C.5 we report the Decrease Acc defined in Equation 3.11 on different activation frequencies γ in our simulated neuron. We say a metric as passing the Theoretical test if it has Decrease Acc $> 90\%$ on every frequency. In tables C.6 and C.7 (visualized in Fig. 3.3) we report the average Δs on the same set of different activation sparsities. We have highlighted metrics with $\Delta s > -0.01$ in red to highlight metrics approaching 0 difference.

We can see most metrics (expect from recall and precision) perform well on balanced data ($\gamma = 0.499$ and $\gamma = 0.1$). However, their performance often starts to drop with score difference Δs approaching 0 as the data becomes more and more unbalanced. We can see that practically all the metrics that failed our experimental missing/extra labels tests cannot differentiate between perfect and modified concept specifically on imbalanced data, highlighting that likely the root cause of the failure on these test is that the metrics performs poorly on imbalanced data. This is also aligned with conventional knowledge that metrics such as accuracy and AUC are a poor choice to rely on when your data is heavily imbalanced. On the other hand, metrics that passed the tests are insensitive to activation frequency γ and converge to a nonzero constant as γ decreases.

The results in terms of passing are almost identical to our experimental results in section 3.4. The only differences were WPMI which passes the theoretical extra labels test but fails

the experimental one. We believe this has to do with hyperparameter(α, λ) choices and that WPMI can in principle pass the test but with poor hyperparameters (small λ) it will not, leading us to overall recommend against using it in practice as hyperparameter choice is challenging in the real world. An interesting case is Inverse AUPRC. This metric is designed for imbalanced data and works well in that domain, but actually performs worse when the data is balanced. In particular Inverse AUPRC fails the test when data is perfectly balanced. This indicates caution should be used if relying on it, in case you have some neurons with extremely common concepts.

We argue that being able to pass these tests regardless of activation frequency is important for any evaluation metric to be used, as we typically do not know what frequency each neuron will have in advance, in many cases the interesting neurons/concept might activate very sparsely, for example in Sparse Autoencoders.

Table C.3. Extended Table (of Table 3.1 in the main text section 3.4) comparing related work. Note [114] call their evaluation Precision, as they use the classification framing. This is equivalent to Recall in our framing as shown in Section C.1.2.

Metric M	Study	Concept Source c_t	Granularity	Probing Dataset $\mathcal{D}$	Domain	Target
Recall	Human Eval [114]	Crowdsourced	$\sim$ Whole Input	Specific eval dataset	Vision	Neurons
$\sim$ Recall	Human Eval [6]	Crowdsourced	$\sim$ Whole Input	Specific eval dataset	Vision	Neurons
$\sim$ Recall	Human Eval [68] [4, 91]	Crowdsourced	Whole Input	Validation data	Vision	Neurons
$\sim$ Recall	Human Eval [67]	Crowdsourced	Whole Input	Validation Data	Vision	CBM neurons
Precision	Automated Eval [91]	Generative	Whole Input	Generated+ Validation data	Vision	Neurons
F1-score	Observation based [42]	Generative + model	Whole Input	Generated + Training data	Language	Neurons
F1-score	Sparse probes [37]	Labeled data	Per-token	Specific eval dataset	Language	Linear comb. of neurons
F1-score	CBM - Concept Error (Fig. 2) [50]	Labeled data	Whole Input	Validation Data	Vision	CBM neurons
IoU	Broden IoU [6] [65, 54]	Labeled data	Per-pixel	Specific eval dataset	Vision	Neurons
Accuracy	CBM - Concept Error [50]	Labeled data	Whole Input	Validation Data	Vision	CBM neurons
$\sim$ AUC	Comparative Human Study [117]	Crowdsourced	Whole Input	Training data	Vision	Neurons
Inverse AUC	INVERT [16]	Labeled data	Whole Input	Validation data	Vision	Neurons
Inverse AUC	CoSy AUC [51]	Generative	Whole Input	Generated+ Validation data	Vision	Neurons
Correlation(T&R)	Simulation - Correlation Score [9]	Model	Per-token	Training data	Language	Neurons
Correlation	Simulation - Correlation Score [69]	Model	Whole Input	Validation data	Vision	Neurons
Correlation	CBM - Concept Error (Fig. 2) [50]	Labeled data	Whole Input	Validation Data	Vision	CBM neurons
Spearman Correlation(T&R)	SAE Auto Interp [14, 96]	Model	Per-token	Training data	Language	SAE features
$\sim$ WPMI	CLIP-Dissect - Similarity [68]	Model	Whole Input	Validation data	Vision	Neurons
MAD	CoSy MAD [51]	Generative	Whole Input	Generated+ Validation data	Vision	Neurons
$\sim$ MAD	MAIA [86]	Generative	Whole Input	Generated	Vision	Neurons
$\sim$ MAD	Explanation Score [88]	Generative	Whole Input	Generated	Language	Transformer factors

Table C.4. Average Decrease Accuracy on Theoretical Missing labels Test. A metric fails if it has low decrease accuracy on any concept frequency γ .

Theoretical Missing Labels Test, Decrease Acc						
Activation Frequency γ :	0.499	0.1	0.01	0.001	0.0001	Pass
Recall	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Precision	0.00%	0.00%	0.00%	0.00%	0.00%	×
F1-score	100.00%	100.00%	100.00%	100.00%	100.00%	✓
IoU	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Accuracy	100.00%	100.00%	100.00%	0.00%	0.00%	×
Balanced Acc.	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Inverse Balanced Acc.	100.00%	100.00%	100.00%	0.00%	0.00%	×
AUC	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Inverse AUC	100.00%	100.00%	100.00%	0.00%	0.00%	×
Correlation	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Correlation(T&R)	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Spearman Correlation	100.00%	100.00%	100.00%	22.70%	13.30%	×
Spearman Correlation(T&R)	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Cosine	100.00%	100.00%	100.00%	100.00%	100.00%	✓
WPMI	100.00%	100.00%	100.00%	100.00%	100.00%	✓
MAD	100.00%	100.00%	100.00%	0.00%	0.00%	×
AUPRC	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Inverse AUPRC	100.00%	100.00%	100.00%	100.00%	100.00%	✓

Table C.5. Average Decrease Accuracy on Theoretical Extra labels Test. A metric fails if it has low decrease accuracy on any concept frequency γ .

Theoretical Extra Labels Test, Decrease Acc						
Activation Frequency γ :	0.499	0.1	0.01	0.001	0.0001	Pass
Recall	0.00%	0.00%	0.00%	0.00%	0.00%	×
Precision	100.00%	100.00%	100.00%	100.00%	100.00%	✓
F1-score	100.00%	100.00%	100.00%	100.00%	100.00%	✓
IoU	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Accuracy	100.00%	100.00%	100.00%	48.40%	0.00%	×
Balanced Acc.	100.00%	100.00%	100.00%	0.00%	0.00%	×
Inverse Balanced Acc.	100.00%	100.00%	100.00%	100.00%	100.00%	✓
AUC	100.00%	100.00%	100.00%	0.00%	0.00%	×
Inverse AUC	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Correlation	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Correlation(T&R)	100.00%	92.80%	22.60%	2.70%	0.10%	×
Spearman Correlation	100.00%	100.00%	5.60%	3.20%	11.60%	×
Spearman Correlation(T&R)	0.20%	8.60%	20.60%	63.70%	5.30%	×
Cosine	100.00%	100.00%	100.00%	100.00%	100.00%	✓
WPMI	100.00%	100.00%	100.00%	100.00%	100.00%	✓
MAD	100.00%	100.00%	100.00%	100.00%	100.00%	✓
AUPRC	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Inverse AUPRC	47.70%	100.00%	100.00%	100.00%	100.00%	×

Table C.6. Simulation results of missing labels test on idealized neuron with perfect correspondence to a concept activation. We can see most metrics pass when the data is relatively balanced, but several start to struggle on imbalanced data (low γ). In contrast, passing metrics maintain a clearly nonzero Δs regardless of activation frequency. The $\lim_{\gamma \rightarrow 0}$ column is analytical solution from C.9.

Theoretical Missing Labels Test, Average Δs						
Activation Frequency γ :	0.499	0.1	0.01	0.001	0.0001	$\lim_{\gamma \rightarrow 0}$
Recall	-0.5000	-0.4999	-0.5002	-0.5007	-0.5025	-0.5000
Precision	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
F1-score	-0.3334	-0.3333	-0.3335	-0.3341	-0.3352	-0.3333
IoU	-0.5000	-0.5002	-0.4998	-0.5005	-0.5032	-0.5000
Accuracy	-0.2495	-0.0500	-0.0050	-0.0005	0.0000	0.0000
Balanced Acc.	-0.2500	-0.2500	-0.2501	-0.2500	-0.2500	-0.2500
Inverse Balanced Acc.	-0.1662	-0.0263	-0.0025	-0.0002	0.0000	0.0000
AUC	-0.2500	-0.2500	-0.2500	-0.2493	-0.2508	-
Inverse AUC	-0.1662	-0.0263	-0.0025	-0.0003	0.0000	-
Correlation	-0.2111	-0.1559	-0.1474	-0.1466	-0.1479	-
Correlation(T&R)	-0.4965	-0.4894	-0.4649	-0.3928	-0.2449	-
Spearman Correlation	-0.2000	-0.0686	-0.0079	-0.0006	-0.0018	-
Spearman Correlation(T&R)	-0.1005	-0.1009	-0.1066	-0.0902	-0.2006	-
Cosine	-0.1464	-0.1465	-0.1465	-0.1464	-0.1474	-
WPMI	-0.3590	-0.3591	-0.3589	-0.3587	-0.3602	-
MAD	-0.2210	-0.0350	-0.0033	-0.0003	0.0000	-
AUPRC	-0.2505	-0.4499	-0.4953	-0.5003	-0.4964	-
Inverse AUPRC	-0.5000	-0.4999	-0.5002	-0.4988	-0.4996	-

Table C.7. Simulation results of extra labels test on idealized neuron with perfect correspondence to a concept activation. We can see most metrics pass when the data is relatively balanced, but several start to struggle on imbalanced data (low γ). In contrast, passing metrics maintain a clearly nonzero Δs regardless of activation frequency. The $\lim_{\gamma \rightarrow 0}$ column is analytical solution from C.9.

Theoretical Extra Labels Test, Average Δs						
Activation Frequency γ :	0.499	0.1	0.01	0.001	0.0001	$\lim_{\gamma \rightarrow 0}$
Recall	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Precision	-0.5000	-0.5001	-0.4999	-0.4993	-0.4963	-0.5000
F1-score	-0.3333	-0.3334	-0.3334	-0.3336	-0.3333	-0.3333
IoU	-0.5000	-0.5000	-0.5001	-0.4997	-0.4970	-0.5000
Accuracy	-0.4990	-0.1000	-0.0100	-0.0010	-0.0001	0.0000
Balanced Acc.	-0.4980	-0.0556	-0.0051	-0.0005	0.0000	0.0000
Inverse Balanced Acc.	-0.2500	-0.2500	-0.2500	-0.2500	-0.2483	-0.2500
AUC	-0.7562	-0.0844	-0.0077	-0.0008	-0.0001	-
Inverse AUC	-0.2500	-0.2500	-0.2501	-0.2500	-0.2491	-
Correlation	-0.4777	-0.1667	-0.1483	-0.1465	-0.1461	-
Correlation(T&R)	-0.4929	-0.0293	-0.0027	-0.0002	0.0000	-
Spearman Correlation	-0.5672	-0.0238	-0.0007	-0.0002	-0.0018	-
Spearman Correlation(T&R)	0.0002	0.0000	-0.0036	0.0307	-0.0064	-
Cosine	-0.1464	-0.1464	-0.1463	-0.1464	0.1456	-
WPMI	-0.0292	-0.0292	-0.0292	-0.0293	-0.0292	-
MAD	-0.3324	-0.3324	-0.3324	-0.3321	-0.3298	-
AUPRC	-0.5000	-0.5000	-0.4999	-0.4998	-0.4974	-
Inverse AUPRC	-0.0010	-0.4000	-0.4899	-0.4984	-0.4957	-

C.4 Analytical solution for Missing/Extra Labels Test

In this section, we provide a theoretical analysis for missing label and extra label test for binary classification metrics. For simplicity of symbols, in this section we analyze the population statistics. Suppose we have following confusion matrix:

	c=1	c=0
a=1	γ	b
a=0	η	d

Here, we use a to denote neuron activation and c to denote concept activation. In missing labels test, consider a general case where we randomly flip $c = 1$ into $c = 0$ with probability p . Thus, the resulting confusion matrix is:

	c=1	c=0
a=1	$(1-p)\gamma$	$b + p\gamma$
a=0	$(1-p)\eta$	$d + p\eta$

In extra labels test, similarly, we turn $c = 0$ into $c = 1$ with probability $q = \frac{p(\gamma+\eta)}{b+d}$, the resulting confusion matrix is

	c=1	c=0
a=1	$\gamma + qb$	$(1-q)b$
a=0	$\eta + qd$	$(1-q)d$

With these, we could plug in corresponding TP/FP/TN/FN into metrics to calculate metric value in these two tests.

1. Recall:

$$s_M(a_k, c_t) = \frac{B(a_k) \cdot B(c_t)}{\|B(a_k)\|_1} = \frac{\gamma}{b + \gamma}. \quad (\text{C.18})$$

In extra label test:

$$s_M(a_k, c_t^+) = \frac{\gamma + qb}{b + \gamma} \geq s_M(a_k, c_t). \quad (\text{C.19})$$

In missing label test:

$$s_M(a_k, c_t^-) = \frac{\gamma - p\gamma}{b + \gamma} \leq s_M(a_k, c_t). \quad (\text{C.20})$$

From the derivation above, we could see that increasing labels only raises recall metric while reducing labels always leads to a drop in recall as we found in our experiments.

2. Precision:

$$s_M(a_k, c_t) = \frac{B(a_k) \cdot B(c_t)}{\|B(c_t)\|_1} = \frac{\gamma}{\gamma + \eta} \quad (\text{C.21})$$

In extra label test:

$$s_M(a_k, c_t^+) = \frac{\gamma + qb}{\gamma + qb + \eta + qd}. \quad (\text{C.22})$$

Precision will increase if $\frac{b}{b+d} > \frac{\gamma}{\gamma + \eta}$.

In missing label test:

$$s_M(a_k, c_t^-) = \frac{(1-p)\gamma}{(1-p)\gamma + (1-p)\eta} = s_M(a_k, c_t). \quad (\text{C.23})$$

Thus, the precision does not change in the missing labels test.

3. F1-score:

$$s_M(a_k, c_t) = \frac{2\gamma}{2\gamma + \eta + b} \quad (\text{C.24})$$

In extra label test:

$$s_M(a_k, c_t^+) = \frac{2\gamma + 2qb}{2\gamma + qb + \eta + qd + b}. \quad (\text{C.25})$$

F1-score increases if $\frac{2b}{b+d} > \frac{2\gamma}{2\gamma + \eta + b}$.

In missing label test:

$$s_M(a_k, c_t^-) = \frac{2(1-p)\gamma}{2(1-p)\gamma + (1-p)\eta + b + p\gamma} = \frac{2\gamma - 2p\gamma}{2\gamma + (1-p)\eta + b - p\gamma}. \quad (\text{C.26})$$

F1-score decreases in missing label test.

4. IoU:

$$s_M(a_k, c_t) = \frac{B(a_k) \cdot B(c_t)}{\|B(a_k)\|_1 + \|B(c_t)\|_1 - B(a_k) \cdot B(c_t)} = \frac{\gamma}{\gamma + \eta + b}. \quad (\text{C.27})$$

In extra label test:

$$s_M(a_k, c_t^+) = \frac{\gamma + qb}{\gamma + \eta + qd + b}. \quad (\text{C.28})$$

IoU increases if $\frac{b}{d} > \frac{\gamma}{\gamma + \eta + b}$.

In missing label test:

$$s_M(a_k, c_t^-) = \frac{(1-p)\gamma}{(1-p)\gamma + (1-p)\eta + b + p\gamma} = \frac{\gamma - p\gamma}{\gamma + (1-p)\eta + b}. \quad (\text{C.29})$$

Thus, IoU decreases in missing label test.

5. Accuracy:

$$s_M(a_k, c_t) = \frac{B(a_k) \cdot B(c_t) + (\mathbf{1} - B(a_k)) \cdot (\mathbf{1} - B(c_t))}{n} = \gamma + d. \quad (\text{C.30})$$

In extra label test:

$$s_M(a_k, c_t^+) = \gamma + qb + d - qd. \quad (\text{C.31})$$

accuracy increases if $b > d$.

In missing label test:

$$s_M(a_k, c_t^-) = \gamma - p\gamma + d + p\eta. \quad (\text{C.32})$$

Accuracy increases if $\eta > \gamma$.

6. Balanced Accuracy:

$$s_M(a_k, c_t) = \frac{B(a_k) \cdot B(c_t)}{2\|B(a_k)\|_1} + \frac{(\mathbf{1} - B(a_k)) \cdot (\mathbf{1} - B(c_t))}{2\|(1 - B(a_k))\|_1} = \frac{\gamma}{2\gamma + 2b} + \frac{d}{2\eta + 2d}. \quad (\text{C.33})$$

In extra label test:

$$s_M(a_k, c_t^+) = \frac{\gamma + qb}{2\gamma + 2b} + \frac{d - qd}{2\eta + 2d}. \quad (\text{C.34})$$

balanced accuracy increases if $\frac{b}{2\gamma + 2b} > \frac{d}{2\eta + 2d}$.

In missing label test:

$$s_M(a_k, c_t^-) = \frac{\gamma - p\gamma}{2\gamma + 2b} + \frac{d + p\eta}{2\eta + 2d}. \quad (\text{C.35})$$

balanced accuracy increases if $\frac{\gamma}{2\gamma + 2b} < \frac{\eta}{2\eta + 2d}$.

7. Inverse Balanced Accuracy:

$$s_M(a_k, c_t) = \frac{B(a_k) \cdot B(c_t)}{2\|B(c_t)\|_1} + \frac{(\mathbf{1} - B(a_k)) \cdot (\mathbf{1} - B(c_t))}{2\|(1 - B(c_t))\|_1} = \frac{\gamma}{2\gamma + 2\eta} + \frac{d}{2b + 2d}. \quad (\text{C.36})$$

In extra label test:

$$s_M(a_k, c_t^+) = \frac{\gamma + qb}{2\gamma + 2\eta + 2qb + 2qd} + \frac{d - qd}{2b + 2d - 2qd - 2qb}. \quad (\text{C.37})$$

In missing label test:

$$s_M(a_k, c_t^-) = \frac{\gamma - p\gamma}{2\gamma + 2\eta - 2p\gamma - 2p\eta} + \frac{d + p\eta}{2b + 2d + 2p\gamma + 2p\eta}. \quad (\text{C.38})$$

C.4.1 Special case - Theoretical Test

In this section, we consider a special case where the neuron activations perfectly match the concept, i.e. $c \equiv a$. In this case, we have $\eta = b = 0$, $d = 1 - \gamma$. Plugging in those variables,

Table C.8. The evaluation scores for different metrics under missing and extra label tests on an *ideal* neuron whose activations perfectly match the presence of our concept.

Metric	Missing label: $s_M(a_k, c_t^-)$	Extra label: $s_M(a_k, c_t^+)$
Recall	$1 - p$	1
Precision	1	$\frac{1}{1+p}$
F1-score	$\frac{2-2p}{2-p}$	$\frac{2}{2+p}$
IoU	$1 - p$	$\frac{1}{1+p}$
Accuracy	$1 - p\gamma$	$1 - \gamma p$
Balanced Accuracy	$1 - \frac{p}{2}$	$1 - \frac{p\gamma}{2(1-\gamma)}$
Inverse Balanced Accuracy	$\frac{(1-\gamma)}{2(1-\gamma)+2p\gamma} + \frac{1}{2}$	$\frac{2+p}{2(1+p)}$

we get the equations for metric scores under sanity test settings in Table C.8. Further simplifying this by plugging in the p values we typically use we get the simple form for score diff in Table C.9.

Table C.9. Further simplifying from Table C.8 by plugging in p values we typically use in our tests, and $s_M(a_k, c_t) = 1$, we can calculate the theoretical score diff after running our tests for different binary metrics on ideal neurons.

Metric	Missing labels test($p = 0.5$): $s_M(a_k, c_t^-) - s_M(a_k, c_t)$	Extra labels test($p = 1$): $s_M(a_k, c_t^+) - s_M(a_k, c_t)$
Recall	-0.5	0
Precision	0	-0.5
F1-score	$-\frac{1}{3}$	$-\frac{1}{3}$
IoU	-0.5	-0.5
Accuracy	$-\frac{\gamma}{2}$	$-\gamma$
Balanced Accuracy	-0.25	$-\frac{\gamma}{2(1-\gamma)}$
Inverse Balanced Accuracy	$-\frac{\gamma}{2(2-\gamma)}$	-0.25

C.5 Ablations and Extra checks

C.5.1 Increase/decrease Fraction in Missing/Extra labels test

In our standard missing labels we reduce the fraction of positive labels by half, i.e.

$$r_- = \frac{\|c_t^-\|_1}{\|c_t\|_1} = 0.5$$

$$r_+ = \frac{\|c_t^+\|_1}{\|c_t\|_1} = 2$$

In this section we run an ablation on the importance of these specific values by running the test on two other combinations of values: $r_- = 0.75$ and $r_+ = 1.33$ (smaller change) and $r_- = 0.33$ and $r_+ = 3$ (larger change). We report the results on final layer neurons (including superclasses) of ViT-B-16 (trained on ImageNet) in Table C.10. We used ground truth c_t . Overall we can see that these parameters do not impact our qualitative observations, i.e. which metric passes vs which doesn't. While larger changes to r typically increase Decrease Acc, it doesn't affect which metrics approach zero score diff vs which do not, which is what we're mostly interested in. Overall this shows our sanity test is not sensitive to the specific parameter choice but instead reflects overall trends of the metric.

Table C.10. Ablation study on sanity checks varying the magnitude of change in labels. We can see overall our results are consistent and not heavily dependent on the choice of r

	Missing Labels Test			Extra Labels Test		
	$r_- = 0.33$	$r_- = 0.5$	$r_- = 0.75$	$r_+ = 1.33$	$r_+ = 2.0$	$r_+ = 3.0$
Recall	99.55%	95.94%	75.34%	0.00%	0.00%	0.00%
Precision	48.20%	46.09%	49.62%	99.32%	99.62%	99.62%
F1-score	97.07%	95.64%	92.78%	99.62%	99.70%	99.70%
IoU	96.84%	95.34%	92.41%	99.32%	99.62%	99.62%
Accuracy	0.00%	0.00%	0.00%	16.24%	63.08%	99.92%
Balanced Acc.	99.47%	95.86%	75.19%	9.17%	23.98%	63.61%
Inverse Balanced Acc.	48.05%	45.56%	48.35%	99.17%	99.62%	99.92%
AUC	98.65%	95.34%	76.32%	16.99%	30.23%	59.32%
Inverse AUC	24.51%	21.35%	16.47%	99.92%	99.92%	99.92%
Correlation	99.92%	99.92%	99.77%	99.92%	99.92%	99.92%
Correlation(T&R)	99.40%	98.12%	90.68%	43.76%	44.96%	48.12%
Spearman Corr.	57.29%	53.91%	46.92%	37.07%	38.05%	37.67%
Spearman Corr.(T&R)	97.14%	93.61%	82.63%	49.92%	50.75%	52.26%
Cosine	100.00%	100.00%	99.85%	99.85%	99.85%	99.85%
WPMI	99.55%	95.94%	75.34%	99.77%	99.70%	99.77%
MAD	50.83%	49.47%	49.02%	99.70%	99.70%	99.70%
AUPRC	99.47%	99.25%	98.27%	99.32%	99.62%	99.62%
Inverse AUPRC	100.00%	100.00%	99.85%	99.70%	99.62%	99.32%

C.5.2 Failure case of cosine: Adding a constant activation

In this section we discuss a failure case of cosine similarity as an evaluation metric. The main idea is that cosine similarity outputs are not independent of the mean of the neuron’s activation, while all other evaluation metrics are. This causes it to associate neurons with large average activation values with very generic concepts that are almost always active. As an example, we added a constant (1) to all activations of concept neurons in a Concept Bottleneck Model [50] trained on CUB200, and ran our meta-AUPRC test(Sec 3.5). All other evaluation metrics are invariant to this change, but the AUPRC of Cosine drops from 99.07% to 0.93% with this small change, as shown in Table C.11. We believe this is a flaw that points against using cosine similarity, as the average activation of a hidden layer neuron is not functionally important, and could be absorbed into biases of the next layer. Instead we recommend using

Pearson correlation which is identical to cosine similarity after normalizing to mean 0, as we show in Section C.1.4.

Table C.11. Adding a constant to the activation values of all neurons causes the cosine similarity to perform very poorly, while other metrics are unchanged.

CUB200 - CBM concept neurons			
	Original a_k	$a'_k = a_k + 1$	
Metric	AUPRC	AUPRC	Δ AUPRC
Recall	0.3958	0.3958	0.0000
Precision	0.6803	0.6803	0.0000
F1-score/IoU	0.6386	0.6386	0.0000
Accuracy	0.5853	0.5853	0.0000
Balanced Acc.	0.7165	0.7165	0.0000
Inverse Balanced Acc.	0.6628	0.6628	0.0000
AUC	0.7045	0.7045	0.0000
Inverse AUC	0.8956	0.8956	0.0000
Correlation	0.8883	0.8883	0.0000
Correlation(T&R)	0.7170	0.7170	0.0000
Spearman Correlation	0.4233	0.4233	0.0000
Spearman Correlation(T&R)	0.4243	0.4243	0.0000
Cosine	0.8985	0.0443	-0.8542
WPMI	0.7323	0.7323	0.0000
MAD	0.7760	0.7760	0.0000
AUPRC	0.6712	0.6712	0.0000
Inverse AUPRC	0.3975	0.3975	0.0000

C.5.3 Random vs Semantic subsets

Our missing/extra labels tests are a mathematical model of semantic sub/superclasses that can be run without knowledge of the semantics or relationships between concepts. To test this mathematical model, we conducted a version of extra labels test on the ImageNet dataset where instead of randomly sampling c^+ , we used the smallest superclass of the concept according to WordNet hierarchy as c^+ , and a version of missing labels test where we used the largest subclass of the concept as c^- . As shown in new Table C.12, the results are essentially identical to using random sub/supersets, showcasing our random sub/superset is a good model of real semantic relationships for our purposes.

Table C.12. Comparison of c_t^- derived as a random subclass of c_t vs a real semantic subclass of c_t derived from the WordNet hierarchy. The missing labels test was evaluated on all 400 superclass neurons created in the final layer, while the extra labels test was conducted on all 1000 individual class neurons. We can see that from the point of view of passing our tests the random subclass behaves almost exactly as a real semantic subclass.

Metric	ViT-B-16 superclass neurons		ViT-B-16 single class neurons	
	Missing Labels Test		Extra Labels Test	
	Random Subsets	Semantic Subsets	Random Supersets	Semantic Supersets
Recall	96.32%	92.89%	0.00%	0.00%
Precision	44.21%	43.68%	100.00%	99.79%
F1-score	97.37%	91.84%	100.00%	99.79%
IoU	96.84%	91.58%	100.00%	99.79%
Accuracy	44.74%	31.32%	46.11%	71.37%
Balanced Acc.	99.47%	97.37%	0.00%	64.53%
Inverse Balanced Acc.	28.95%	37.37%	100.00%	99.79%
AUC	99.21%	97.11%	17.47%	52.11%
Inverse AUC	51.58%	54.47%	100.00%	96.00%
Correlation	100.00%	100.00%	100.00%	99.58%
Correlation(T&R)	100.00%	99.21%	47.47%	70.11%
Spearman Corr.	70.53%	75.26%	37.37%	20.63%
Spearman Corr.(T&R)	98.42%	97.63%	51.05%	35.16%
Cosine	100.00%	100.00%	100.00%	99.58%
WPMI	100.00%	100.00%	0.00%	0.00%
MAD	58.42%	51.58%	100.00%	100.00%
AUPRC	99.21%	96.84%	100.00%	99.37%
Inverse AUPRC	100.00%	100.00%	100.00%	99.37%

C.5.4 Epsilon ablation

In this section we study the effect of the choice of ε in Equation 3.10 on our experimental missing/extra labels test. Overall we can see in Table C.13 that while changing ε changes results for some metrics, for the most part it does not affect which metrics pass or do not pass the test despite 2 orders of magnitude change in ε , showing our tests are not very sensitive to this choice.

Perhaps more importantly, theoretically we do not think the ε choice is important. An alternative and perhaps more fundamental definition for our theoretical tests could be as follows: A metric passes the test if

$$\exists \varepsilon > 0 \text{ s.t. } \forall \gamma > 0, \Delta s < -\varepsilon \quad (\text{C.39})$$

where Δs is calculated with the theoretical missing labels test defined in Appendix C.3 and γ is the concept/neuron activation frequency. This definition is purely a mathematical property of the metric without any hyperparameters or ties to any setting, and as far as we know matches our theoretical test results of Table C.4 and Table C.5 that use a fixed $\varepsilon = 0.001$.

The issue with this alternative definition is that, we can only prove a metric passes this version of the test if we have an analytical solution, which we have for binary classification metrics in Table C.9. This makes this definition hard to use in practice. We can see for example accuracy fails this test as $\lim_{\gamma \rightarrow 0} \Delta s = 0$, so no matter what ε we choose, $\exists \gamma \text{ s.t. } \Delta s > -\varepsilon$. On the other hand, for F1-score will pass the theoretical tests for any $\varepsilon < 1/3$ regardless of γ . From this we can see that if we decrease ε in the current theoretical test, more metrics would pass, but the same metrics would fail again if we expand Table C.4/C.5 to the right by including smaller values, while current passing metrics would not fail regardless of how many additional γ values we test.

Table C.13. An Ablation study on the effect of ε choice on our results. While the exact percentages vary, we can see that changing ε by orders of magnitude mostly does not change which metrics pass the tests, highlighting robustness to this hyperparameter choice.

Setting 1: ViT-B-16 Final Layer neurons						
	Missing Labels Test - Decrease Acc			Extra Labels Test - Decrease Acc		
	$\varepsilon = 0.0001$	$\varepsilon = 0.001$	$\varepsilon = 0.01$	$\varepsilon = 0.0001$	$\varepsilon = 0.001$	$\varepsilon = 0.01$
Recall	97.07%	97.07%	97.07%	0.00%	0.00%	0.00%
Precision	48.65%	47.14%	37.74%	99.70%	99.20%	97.89%
F1-score	96.48%	95.79%	93.38%	99.77%	99.70%	98.72%
IoU	96.69%	95.41%	92.48%	99.70%	99.62%	97.89%
Accuracy	66.09%	0.00%	0.00%	99.92%	61.50%	6.32%
Balanced Acc.	97.07%	97.07%	97.07%	99.77%	23.08%	3.76%
Inverse Balanced Acc.	51.95%	46.54%	31.35%	99.92%	99.62%	96.77%
AUC	96.17%	95.86%	95.19%	84.81%	29.25%	11.58%
Inverse AUC	78.35%	21.43%	3.68%	99.92%	99.92%	99.92%
Correlation	100.00%	100.00%	99.92%	99.92%	99.92%	99.92%
Correlation(T&R)	97.52%	97.29%	96.47%	48.42%	46.02%	37.07%
Spearman Corr.	63.68%	53.91%	5.94%	48.87%	37.07%	1.28%
Spearman Corr.(T&R)	93.91%	93.76%	92.56%	51.73%	50.90%	44.36%
Cosine	100.00%	100.00%	100.00%	99.92%	99.85%	99.70%
WPMI	97.07%	97.07%	97.07%	99.77%	99.70%	99.55%
MAD	52.78%	50.75%	26.32%	99.70%	99.70%	99.70%
AUPRC	99.77%	99.40%	97.59%	99.70%	99.62%	97.89%
Inverse AUPRC	100.00%	100.00%	100.00%	99.62%	99.62%	99.62%

C.5.5 Number of samples ablation

Most of our results only use one sample of $c_t^\pm$ to reduce computational cost, even though our equations are defined in terms of its expectation. In Table C.14 we test whether using more samples of $c_t^\pm$ affects our results. As we can see, the number of samples has very little effect on our results likely because they are already averages over many neurons and large datasets.

Table C.14. A study measuring whether our test results are sensitive to sampling $c_t^\pm$ only once per neuron. We see this randomness has no effect on our results in terms of which metrics pass and do not pass the test, likely because the results are already averaged over hundreds or thousands of neurons.

Setting 1: ViT-B-16 Final Layer neurons								
$c_t^\pm$ samples per neuron:	Missing Labels Test				Extra Labels Test			
	1	3	10	100	1	3	10	100
Recall	97.07%	100.00%	100.00%	100.00%	0.00%	0.00%	0.00%	0.00%
Precision	47.14%	45.04%	44.29%	38.80%	99.20%	99.62%	99.62%	99.62%
F1-score	95.79%	96.77%	97.29%	97.74%	99.70%	99.70%	99.70%	99.70%
IoU	95.41%	96.02%	96.62%	97.22%	99.62%	99.62%	99.62%	99.62%
Accuracy	0.00%	0.00%	0.00%	0.00%	61.50%	61.88%	64.66%	66.54%
Balanced Acc.	97.07%	99.92%	99.77%	99.85%	23.08%	22.48%	23.31%	22.18%
Inverse Balanced Acc.	46.54%	43.83%	42.41%	34.44%	99.62%	99.62%	99.62%	99.62%
AUC	95.86%	99.25%	99.70%	99.77%	29.25%	26.84%	27.59%	28.35%
Inverse AUC	21.43%	19.17%	16.32%	14.21%	99.92%	99.92%	99.92%	99.92%
Correlation	100.00%	100.00%	99.92%	99.92%	99.92%	99.92%	99.92%	99.92%
Correlation(T&R)	97.29%	99.32%	99.85%	99.77%	46.02%	52.48%	54.51%	55.19%
Spearman Corr.	53.91%	54.14%	54.59%	55.64%	37.07%	34.96%	35.86%	35.04%
Spearman Corr.(T&R)	93.76%	96.39%	97.67%	98.65%	50.90%	57.97%	51.95%	53.83%
Cosine	100.00%	100.00%	100.00%	100.00%	99.85%	99.85%	99.85%	99.85%
WPMI	97.07%	100.00%	100.00%	100.00%	99.70%	99.77%	99.77%	99.85%
MAD	50.75%	48.95%	49.40%	39.17%	99.70%	99.70%	99.70%	99.70%
AUPRC	99.40%	99.32%	99.32%	99.32%	99.62%	99.62%	99.62%	99.62%
Inverse AUPRC	100.00%	100.00%	100.00%	100.00%	99.62%	99.62%	99.62%	99.62%

C.5.6 Combination Metrics

In addition to individual metrics we studied in the main paper, we also explore whether combinations of different metrics could serve as good evaluation metrics. This investigation is inspired by the success of F1-score, which is the harmonic mean of the *Recall* and *Precision* metrics.

$$s_{F1} = \frac{2}{1/s_{\text{Recall}} + 1/s_{\text{Precision}}} = \frac{2s_{\text{Recall}}s_{\text{Precision}}}{s_{\text{Recall}} + s_{\text{Precision}}} \quad (\text{C.40})$$

In this section we investigated using the harmonic mean of other metrics as our metric. The missing and extra labels test results are reported in Tables C.15, C.16 and C.17. We can see multiple combinations perform well and now pass the theoretical missing/extra labels tests. In general, our initial results indicate that combining a metric that passes the extra labels test with a metric that passes the missing labels test will pass both tests most of the time. Conversely, combining two metrics that fail the same test, such as Recall and AUC will still fail that test. In table C.18, we also measure the AUPRC (Sec 3.5) performance of these new metrics on a subset of our settings, and show that many combination metrics are competitive with top standard metrics. In particular the harmonic mean of Balanced Acc and Inverse Balanced Acc performed well. Overall this shows the promise of using novel combinations of concepts, but a more thorough study is needed to confirm their reliability.

Table C.15. Theoretical Missing Labels test on combination metrics.

Theoretical Missing Labels Test, Decrease Acc						
Activation Frequency γ :	0.499	0.1	0.01	0.001	0.0001	Pass
AUC + Inverse AUC	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Balanced Acc + Inverse Balanced Acc	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Recall + AUC	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Recall + Inverse AUC	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Precision + Balanced Acc	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Precision + Inverse Balanced Acc	100.00%	100.00%	100.00%	0.00%	0.00%	×
F1-score	100.00%	100.00%	100.00%	100.00%	100.00%	✓

Table C.16. Theoretical Extra Labels Test for combination metrics.

Theoretical Extra Labels Test, Decrease Acc						
Activation Frequency γ :	0.499	0.1	0.01	0.001	0.0001	Pass
AUC + Inverse AUC	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Balanced Acc + Inverse Balanced Acc	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Recall + AUC	100.00%	100.00%	100.00%	0.00%	0.00%	×
Recall + Inverse AUC	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Precision + Balanced Acc	100.00%	100.00%	100.00%	100.00%	100.00%	✓
Precision + Inverse Balanced Acc	100.00%	100.00%	100.00%	100.00%	100.00%	✓
F1-score	100.00%	100.00%	100.00%	100.00%	100.00%	✓

Table C.17. Experimental Missing and Extra Labels Test with new combination metrics.

Combination Metric	Setting 1: ViT-B-16 final layer		Setting 2: Resnet-50 layer4	
	Missing Labels	Extra Labels	Missing Labels	Extra Labels
	Decrease acc	Decrease acc	Decrease acc	Decrease acc
AUC + Inverse AUC	95.64%	99.77%	89.05%	99.90%
Balanced Acc + Inverse Balanced Acc	99.85%	99.92%	90.75%	100.00%
Recall + AUC	96.99%	19.77%	100.00%	17.32%
Recall + Inverse AUC	96.92%	99.92%	100.00%	87.00%
Precision + Balanced Acc	91.80%	99.70%	64.39%	100.00%
Precision + Inverse Balanced Acc	48.87%	99.92%	47.38%	100.00%
F1-score	95.86%	99.70%	100.00%	100.00%

Table C.18. Performance of new combination on the meta-AUPRC evaluation introduced in section 3.5. We can see some combination metrics such as Balanced Acc + Inverse Balanced Acc perform well.

	Setting 3: gt c_t		Setting 4: SigLIP c_t	
	ViT-B-16(ImageNet)		ViT-B-16(ImageNet)	
New metrics:	AUPRC	Rank	AUPRC	Rank
AUC + Inverse AUC	0.8977	3	0.6551	10
Balanced Acc + Inverse Balanced Acc	0.8885	5	0.7041	3
Recall + AUC	0.7081	17	0.6129	15
Recall + Inverse AUC	0.8892	4	0.0077	22
Precision + Balanced Acc	0.8679	8	0.7141	2
Precision + Inverse Balanced Acc	0.8610	9	0.6468	12
Old metrics:				
Recall	0.0832	22	0.5758	17
Precision	0.8592	11	0.6428	13
F1-score/IoU	0.8759	7	0.6992	5
Accuracy	0.7858	13	0.5853	16
Balanced Accuracy	0.7463	15	0.6158	14
Inverse Balanced Acc.	0.8610	9	0.6474	11
AUC	0.7139	16	0.6933	7
Inverse AUC	0.8120	12	0.5439	18
Correlation	0.9399	1	0.7027	4
Correlation(T&R)	0.4365	18	0.4906	19
Spearman Correlation	0.0125	23	0.0047	23
Spearman Correlation(T&R)	0.1783	20	0.2395	20
Cosine	0.9037	2	0.6948	6
WPMI	0.7570	14	0.6881	8
MAD	0.1866	19	0.1268	21
AUPRC	0.8839	6	0.7511	1
Inverse AUPRC	0.1131	21	0.6688	9

C.6 Detailed results

C.6.1 Missing and Extra Labels Test:

Experimental Setup Details:

We evaluate our experimental results across 8 settings:

- **Setting 1:** Dataset $\mathcal{D}$ =ImageNet, ViT-B-16 [20] trained on ImageNet, 1400 final layer neurons(and superclass neurons), Table C.19
- **Setting 2:** Dataset $\mathcal{D}$ =ImageNet, ResNet-50 trained on ImageNet, 2048 layer4 neurons, Table C.19
- **Setting 3:** Dataset $\mathcal{D}$ =Places365, ResNet-18 trained on Places365, 365 final layer neurons, Table C.20
- **Setting 4:** Dataset $\mathcal{D}$ =Places365, ResNet-18 trained on Places365, 512 layer4 neurons, Table C.20
- **Setting 5:** Dataset $\mathcal{D}$ =CUB200, CBM trained on CUB200, 112 concept neurons, Table C.21
- **Setting 6:** Dataset $\mathcal{D}$ =CUB200, CLIP ViT-B-32 image encoder, linear probe trained to detect CUB-concepts, 112 concepts, Table C.21
- **Setting 7:** Dataset $\mathcal{D}$ =OpenWebText(subset), GPT-2-small, final(prediction) layer neurons corresponding to 500 most common tokens, Table C.22
- **Setting 8:** Dataset $\mathcal{D}$ =OpenWebText(subset), GPT-2-XL, final(prediction) layer neurons corresponding to 500 most common tokens, Table C.22

For all settings we used the ground truth labels from the dataset as c_t . The ImageNet [19], Places [115] and GPT-2 [78] models were pretrained. For CUB-CBM we trained our own model using the code released by [50]. Our CBM reached 96.75% concept accuracy on the test

set which is in line with their reported results. For CLIP, we used the pretrained model from [77], and then learned a linear probe on top of frozen image embeddings to minimize binary cross-entropy loss on the training split of CUB200[98], with early stopping using validation data. Our linear probe reached 89.76% concept accuracy. The CUB dataset is a small bird species classification dataset that contains detailed annotations for lower level concepts, such as wing color. Following [50], we only used the 112 concepts that are present on at least 5% of the inputs and our CLIP linear probe was trained to predict these concepts, not the final class of inputs.

For the final layer neurons as well as CUB neurons we let "correct" concept t_k be the ground truth concept for that neuron. We choose the hyperparameter α that maximizes AUPRC(Sec 3.5) performance on validation neurons, and run the tests on test neurons. For all evaluations we used neuron activations after the activation function (i.e. softmax/sigmoid).

For the Language Model evaluations we used a subset of 204,800 tokens from Open-WebText [32] as $\mathcal{D}$. We evaluated final layer neurons that directly predict next-token, so their explanation was a specific token, which allowed us to extract ground truth c_t directly from the text. We focused on the neurons predicting 500 most common tokens to reduce computational cost and to make sure we focus on concepts where there are enough positive labels available for statistical significance.

For layer4(after avg pool) neurons we defined the "correct" concept t_k as the concept that maximizes IoU with $\alpha = 0.005$ similar to [6], using the class(and superclass) labels of the dataset as c_t . For these layers we fixed $\alpha = 0.005$ for all metrics as that was used to determine the "ground truth".

Interestingly, we find that more methods pass the tests on the CUB dataset than the other datasets we looked at, but the trends in terms of which metrics perform worse are still similar. We believe this is caused by data imbalance, as the concepts in CUB are relatively balanced (following [50] we only keep concepts that are present on at least 5% of the inputs), while for example ImageNet classes are much more imbalanced (each class is positive on 0.1% of the inputs). This is confirmed by our theoretical observations in Section C.3, which show that poor

metrics are much more likely to fail the test when the concepts are imbalanced.

Table C.19. Detailed results of Missing/Extra Labels test on Setting 1 and 2.

Metric	Setting 1: ViT-B-16 final layer neurons		Setting 2: Resnet-50 layer4 neurons	
	Missing Labels Test	Extra Labels Test	Missing Labels Test	Extra Labels Test
	Decrease Acc	Decrease Acc	Decrease Acc	Decrease Acc
Recall	95.94%	0.00%	100.00%	0.00%
Precision	46.09%	99.62%	47.33%	100.00%
F1-score	95.64%	99.70%	100.00%	100.00%
IoU	95.34%	99.62%	100.00%	100.00%
Accuracy	0.00%	63.08%	0.00%	71.07%
Balanced Acc.	95.86%	23.98%	100.00%	21.12%
Inverse Balanced Acc.	45.56%	99.62%	46.35%	100.00%
AUC	95.34%	30.23%	87.41%	49.02%
Inverse AUC	21.35%	99.92%	46.61%	100.00%
Correlation	99.92%	99.92%	100.00%	100.00%
Correlation(T&R)	98.12%	44.96%	85.05%	50.51%
Spearman Corr.	53.91%	38.05%	56.89%	38.08%
Spearman Corr.(T&R)	93.61%	50.75%	67.47%	48.72%
Cosine	100.00%	99.85%	100.00%	97.84%
WPMI	95.94%	99.70%	86.64%	100.00%
MAD	49.47%	99.70%	46.30%	100.00%
AUPRC	99.25%	99.62%	99.23%	100.00%
Inverse AUPRC	100.00%	99.62%	98.05%	98.05%

Table C.20. Detailed results of Missing/Extra Labels test on Setting 3 and 4.

Metric	Setting 3: RN18(Places365) final layer		Setting 4: RN18(Places365) layer4	
	Missing Labels Test	Extra Labels Test	Missing Labels Test	Extra Labels Test
	Decrease Acc	Decrease Acc	Decrease Acc	Decrease Acc
Recall	95.68%	0.00%	100.00%	0.00%
Precision	48.13%	98.85%	45.38%	100.00%
F1-score	56.77%	98.85%	99.79%	100.00%
IoU	56.77%	98.85%	99.79%	100.00%
Accuracy	0.00%	100.00%	0.00%	100.00%
Balanced Acc.	95.68%	99.42%	100.00%	65.71%
Inverse Balanced Acc.	44.67%	98.85%	45.17%	100.00%
AUC	99.14%	59.37%	87.89%	51.33%
Inverse AUC	41.50%	100.00%	50.31%	100.00%
Correlation	99.14%	99.42%	100.00%	100.00%
Correlation(T&R)	95.10%	48.70%	83.16%	51.33%
Spearman Corr.	60.23%	37.46%	64.07%	44.76%
Spearman Corr.(T&R)	89.91%	51.59%	70.43%	50.51%
Cosine	99.42%	99.42%	100.00%	99.79%
WPMI	95.68%	0.00%	91.17%	100.00%
MAD	46.97%	99.71%	45.38%	100.00%
AUPRC	84.44%	97.98%	97.13%	100.00%
Inverse AUPRC	100.00%	99.71%	99.18%	100.00%

Table C.21. Detailed results of Missing/Extra Labels test on Setting 5 and 6.

Setting 5: CUB200 CBM neurons			Setting 6: CUB200 Linear Probe	
	Missing Labels Test	Extra Labels Test	Missing Labels Test	Extra Labels Test
Metric	Decrease Acc	Decrease Acc	Decrease Acc	Decrease Acc
Recall	100.00%	0.00%	100.00%	0.00%
Precision	38.32%	100.00%	43.93%	100.00%
F1-score	100.00%	100.00%	100.00%	100.00%
IoU	100.00%	100.00%	100.00%	100.00%
Accuracy	86.92%	100.00%	100.00%	95.33%
Balanced Acc.	100.00%	100.00%	100.00%	100.00%
Inverse Balanced Acc.	100.00%	100.00%	98.13%	100.00%
AUC	100.00%	100.00%	100.00%	90.65%
Inverse AUC	100.00%	100.00%	100.00%	100.00%
Correlation	100.00%	100.00%	100.00%	100.00%
Correlation(T&R)	99.07%	93.46%	98.13%	94.39%
Spearman Corr.	100.00%	90.65%	100.00%	87.85%
Spearman Corr.(T&R)	93.46%	89.72%	95.33%	85.05%
Cosine	100.00%	100.00%	100.00%	97.20%
WPMI	100.00%	91.59%	100.00%	79.44%
MAD	95.33%	99.07%	96.26%	96.26%
AUPRC	100.00%	100.00%	100.00%	100.00%
Inverse AUPRC	100.00%	93.46%	100.00%	73.83%

Table C.22. Detailed results of Missing/Extra Labels test on Setting 7 and 8.

Setting 7: GPT-2-small final layer			Setting 8: GPT-2-XL final layer	
	Missing Labels Test	Extra Labels Test	Missing Labels Test	Extra Labels Test
Metric	Decrease Acc	Decrease Acc	Decrease Acc	Decrease Acc
Recall	98.11%	0.00%	99.58%	0.00%
Precision	47.16%	100.00%	49.47%	100.00%
F1-score	99.37%	100.00%	97.89%	100.00%
IoU	99.16%	100.00%	97.89%	100.00%
Accuracy	1.47%	16.63%	1.89%	16.84%
Balanced Acc.	98.11%	9.68%	99.58%	9.47%
Inverse Balanced Acc.	45.47%	100.00%	48.84%	99.16%
AUC	93.89%	43.79%	96.00%	49.05%
Inverse AUC	34.11%	100.00%	28.63%	100.00%
Correlation	97.47%	100.00%	98.74%	100.00%
Correlation(T&R)	69.26%	48.00%	74.74%	50.74%
Spearman Corr.	38.53%	26.95%	38.74%	29.89%
Spearman Corr.(T&R)	65.05%	52.21%	65.05%	49.89%
Cosine	97.47%	100.00%	98.74%	100.00%
WPMI	98.11%	0.00%	99.58%	0.00%
MAD	49.68%	100.00%	49.05%	100.00%
AUPRC	91.16%	98.32%	93.68%	99.79%
Inverse AUPRC	97.68%	100.00%	98.32%	100.00%

C.6.2 Meta-AUPRC evaluation:

Experimental Setup Details:

We evaluate our meta-AUPRC test described in Section 3.5 on 10 different settings:

- **Setting 1:** Dataset $\mathcal{D}$ =ImageNet, ViT-B-16(ImageNet), 1000 final layer neurons, ground truth c_t
- **Setting 2:** Dataset $\mathcal{D}$ =ImageNet, ViT-B-16(ImageNet), 1000 final layer neurons, SigLIP c_t
- **Setting 3:** Dataset $\mathcal{D}$ =ImageNet, ViT-B-16(ImageNet), 1400 final layer neurons + super-class neurons, ground truth c_t
- **Setting 4:** Dataset $\mathcal{D}$ =ImageNet, ViT-B-16(ImageNet), 1400 final layer neurons + super-class neurons, SigLIP c_t
- **Setting 5:** Dataset $\mathcal{D}$ =Places365, ResNet-18(Places365), 365 final layer neurons, ground truth c_t
- **Setting 6:** Dataset $\mathcal{D}$ =Places365, ResNet-18(Places365), 365 final layer neurons, SigLIP c_t
- **Setting 7:** Dataset $\mathcal{D}$ =CUB200, CBM trained on CUB200, 112 concept neurons, ground truth c_t
- **Setting 8:** Dataset $\mathcal{D}$ =CUB200, CLIP ViT-B-32 image encoder, linear probe trained to detect 112 CUB-concepts, ground truth c_t
- **Setting 9:** Dataset $\mathcal{D}$ =OpenWebText(subset), GPT-2-small, final(prediction) layer neurons corresponding to 500 most common tokens.
- **Setting 10:** Dataset $\mathcal{D}$ =OpenWebText(subset), GPT-2-XL, final(prediction) layer neurons corresponding to 500 most common tokens.

SigLIP c_t indicates we used Pseudo-labels generated from SigLIP (ViT-SO400M-14-SigLIP-384) [111] as done by [69]. For all metrics and evaluations we choose hyperparameters such as α by finding the one with best performance on validation neurons (random subset of 5% of the neurons), and use those hyperparameters to evaluate on test neurons.

Tables C.23, C.24 and C.25 show the detailed results of our AUPRC evaluation experiment. While there is some differences on which metrics perform well on different setups, the overall trends are quite consistent, with Correlation and Cosine outperforming others on almost all settings, and AUPRC being the third best. Spearman Correlation also consistently performed the worst.

Table C.23. Meta-AUPRC evaluation results on ImageNet Settings(1-4).

	Setting 1: gt c_t		Setting 2: SigLIP c_t		Setting 3: gt c_t		Setting 4: SigLIP c_t	
	original K and C		original K and C		original+superclass K and C		original+superclass K and C	
Metric	AUPRC	Rank	AUPRC	Rank	AUPRC	Rank	AUPRC	Rank
Recall	0.9989	6	0.8638	12	0.0832	16	0.5758	12
Precision	0.9989	6	0.9640	6	0.8592	6	0.6428	9
F1-score/IoU	0.9989	6	0.9477	7	0.8759	4	0.6992	3
Accuracy	0.9989	6	0.8497	13	0.7858	8	0.5853	11
Balanced Acc.	0.9989	6	0.9032	11	0.7463	10	0.6158	10
Inverse Balanced Acc.	0.9989	6	0.9664	5	0.8610	5	0.6474	8
AUC	0.9974	14	0.9438	8	0.7139	11	0.6933	5
Inverse AUC	0.9415	15	0.8452	14	0.8120	7	0.5439	13
Correlation	0.9998	<u>2</u>	<u>0.9739</u>	<u>2</u>	0.9399	1	<u>0.7027</u>	<u>2</u>
Correlation(T&R)	0.9993	5	0.7257	15	0.4365	12	0.4906	14
Spearman Corr.	0.0023	17	0.0213	17	0.0125	17	0.0047	17
Spearman Corr.(T&R)	0.6533	16	0.4100	16	0.1783	14	0.2395	15
Cosine	<u>0.9998</u>	<u>2</u>	0.9733	3	<u>0.9037</u>	<u>2</u>	0.6948	4
WPMI	0.9989	6	0.9211	9	0.7570	9	0.6881	6
MAD	0.9994	4	0.9716	4	0.1866	13	0.1268	16
AUPRC	0.9989	6	0.9866	1	0.8839	3	0.7511	1
Inverse AUPRC	0.9999	1	0.9203	10	0.1131	15	0.6688	7

Table C.24. Meta-AUPRC evaluation results on Places and CUB models Settings(5-8).

	Setting 5: gt c_t		Setting 6: SigLIP c_t		Setting 7: gt c_t		Setting 8: gt c_t	
	Resnet-18(Places)		Resnet-18(Places)		CBM(CUB200)		Linear Probe(CUB200)	
Metric	AUPRC	Rank	AUPRC	Rank	AUPRC	Rank	AUPRC	Rank
Recall	0.9639	12	0.7623	9	0.3958	17	0.1078	17
Precision	0.9640	6	0.7771	8	0.6803	9	0.2095	12
F1-score/IoU	0.9640	6	0.7916	6	0.6386	12	0.2792	7
Accuracy	0.9640	6	0.6490	14	0.5853	13	0.2239	10
Balanced Acc.	0.9640	6	0.7340	11	0.7165	7	0.3186	4
Inverse Balanced Acc.	0.9640	6	0.7813	7	0.6628	11	0.2608	8
AUC	0.9418	14	0.7205	12	0.7045	8	0.2031	14
Inverse AUC	0.8927	15	0.6877	13	<u>0.8956</u>	<u>2</u>	0.3745	3
Correlation	0.9693	1	<u>0.8326</u>	<u>2</u>	0.8883	3	0.4683	1
Correlation(T&R)	0.9662	4	<u>0.5326</u>	<u>15</u>	0.7170	6	0.2976	6
Spearman Corr.	0.0213	17	0.1214	17	0.4233	15	0.2185	11
Spearman Corr.(T&R)	0.7424	16	0.4005	16	0.4243	14	0.1607	15
Cosine	0.9693	1	0.8304	4	0.8985	1	<u>0.4067</u>	<u>2</u>
WPMI	0.9655	5	0.7539	10	0.7323	5	0.2089	13
MAD	0.9590	13	0.8321	3	0.7760	4	0.2432	9
AUPRC	0.9640	6	0.8669	1	0.6712	10	0.2981	5
Inverse AUPRC	0.9678	3	0.8006	5	0.3975	16	0.1302	16

Table C.25. Meta-AUPRC evaluation results on Language Models Settings(9-10).

	Setting 9: gt c_t		Setting 10: gt c_t	
	GPT2-small		GPT2-XL	
Metric	AUPRC	Rank	AUPRC	Rank
Recall	0.9754	6	0.9951	6
Precision	0.9655	8	0.9780	9
F1-score/IoU	0.9569	9	0.9875	7
Accuracy	0.7395	13	0.8334	14
Balanced Acc.	0.9842	4	0.9973	4
Inverse Balanced Acc.	0.9658	7	0.9782	8
AUC	0.8139	12	0.9194	12
Inverse AUC	0.7282	14	0.8481	13
Correlation	0.9918	1	0.9986	1
Correlation(T&R)	0.6705	15	0.7695	15
Spearman Correlation	0.0137	17	0.0142	17
Spearman Correlation(T&R)	0.0917	16	0.1168	16
Cosine	<u>0.9912</u>	<u>2</u>	<u>0.9985</u>	<u>2</u>
WPMI	<u>0.9773</u>	<u>5</u>	<u>0.9958</u>	<u>5</u>
MAD	0.9014	11	0.9557	11
AUPRC	0.9880	3	0.9951	6
Inverse AUPRC	0.9314	10	0.9780	9

Appendix D

Supplementary Material for Chapter 4

D.1 Discussion

D.1.1 Limitations

Due to time and budget constraints, our crowdsourced evaluation focuses on comparing descriptions on a few models/layers e.g. later layer neurons of ImageNet trained models. While we mostly find similar trends between the ResNet and ViT models, different description methods may have advantage on different types of neurons. For example, it is possible that Network Dissection based methods would perform better than they did on our current evaluation if we focus on lower layer neurons or on models trained on Places365 [115], as the labels in the Broden dataset are more suitable for these tasks. Similarly, we think LLM based methods [4, 86] might perform better when describing neurons of a sparse autoencoder [14] as these are more monosemantic and can be better described by highly activating inputs only.

Second, our crowdsourced evaluation relies on Amazon Mechanical Turk workers, who are not experts and often make errors in labeling. While we introduced principled measures to estimate the errors as well as mitigations for error, we cannot improve their domain knowledge, which means a crowdsourced evaluation might favor simpler descriptions over more complex concepts requiring domain knowledge. To compare more complex descriptions or neuron descriptions in a specific domain it may be necessary to recruit domain expert raters.

Similarly, our Mechanical Turk validation in Section 4.4 assumes that the ImageNet

labels for these images are correct and unambiguous, but in reality the labels contain some errors and ambiguity. To mitigate this, we focused on ImageNet classes we deemed to have less ambiguity in our experiment, but some ambiguity still remains as discussed in Appendix D.3.7. Using an alternate rating setup that allows for rating indeterminacy such as [34] could potentially improve this validation.

Finally, our evaluation is focused on evaluating *input-based* neuron explanations that aim to explain the "input $\rightarrow$ neuron activation" function. Some recent work such as [28, 36] instead focus on *output-based* neuron explanations that aim to explain the "neuron activation $\rightarrow$ model output" relationship. Rigorously evaluating these *output-based* explanations will require different methodology and is an interesting problem for future work.

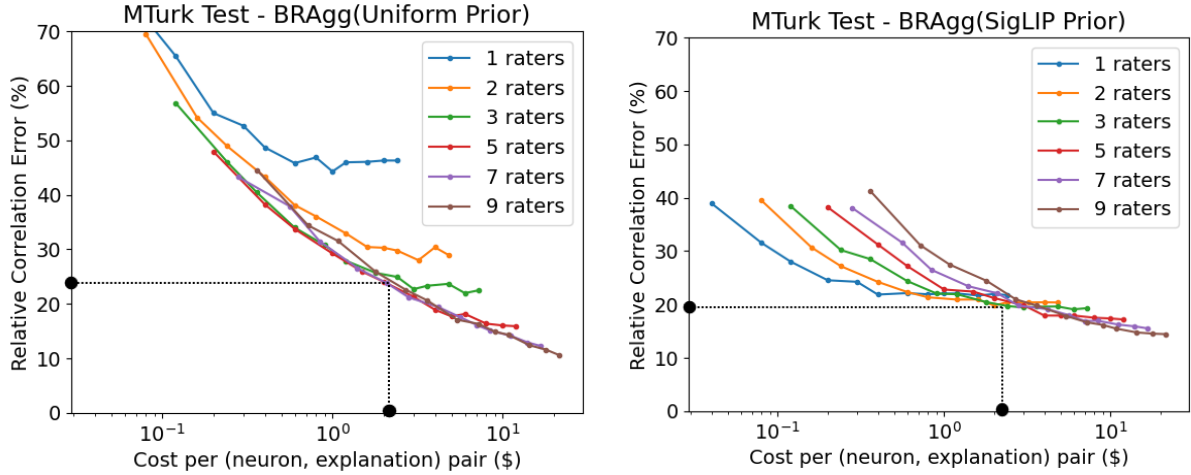
D.1.2 Broader Impact

This paper is focused on better understanding neural networks via interpretability, and as such we expect its impact to be largely positive as better understanding of neural networks can help us for example identify failure modes before deploying and enable better control of models. As our focus is in particular on rigorous evaluation of neuron explanations, this can help avoid interpretability illusions or users over-relying on unreliable explanations.

D.1.3 Crowdsourced Study Design

In this section we discuss how our methodology testing/validation settings described in Section 4.4 can be used to help design the finer parameters of our study such as how many raters to use per input. In our current MTurk setup, the cost of obtaining a rating for a single image by a single rater is $\frac{\$0.06}{15} = \0.004 as we show 15 inputs per task. We can then plot the expected error as a function of evaluation cost by plotting $n_{inputs} \cdot n_{raters} \cdot \0.004 on the x-axis.

In Figure D.1, we plot the cost vs expected error rate for different numbers of raters we have per input. As we can see, the optimal number of raters per input depends on your budget, with smaller number of raters working better with smaller budgets, while for a larger budget it is



(a) Aggregation with Method 3a: BRAGg - Uniform Prior.

(b) Aggregation with Method 3b: BRAGg - SigLIP prior.

Figure D.1. Comparing rating aggregation strategies in Setting 2 (Testing on MTurk) described in Sec 4.4. The x -axis represents the cost of evaluations as number of inputs per neuron multiplied by number of raters per input, times cost per input. The lowest curve represents the optimal number of raters for a particular budget.

optimal to use more raters per input.

Given our budget, we are aiming for a cost of around \$2.2 per (neuron, explanation pair), and we can see the optimal number of raters and expected error rate for this budget following the black lines on our plot. Since we wish to get good results using both aggregation methods, we can see 3 raters (green line) is among the best for both methods, we choose to use 3 raters and 180 inputs per neuron for the crowd-sourced study. This gives us an expected correlation error of around 25% with Bayes - Uniform Prior and around 20% with Bayes - SigLIP prior based on the MTurk test.

D.2 Ablation Studies

D.2.1 Choice of γ

Figure D.2 shows the effects of the hyperparameter γ (Eq. 4.6) on importance sampling performance on our simulated setup with no label noise. For visual clarity, we report the difference compared to baseline with $\gamma = 0$ instead of absolute numbers, so for example reducing

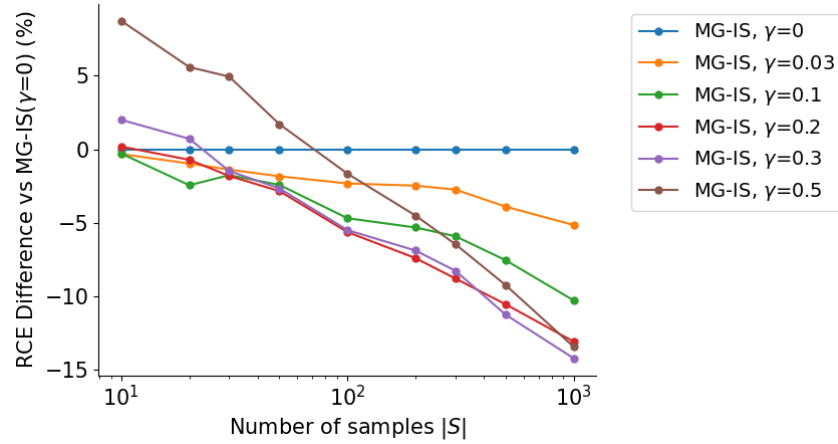


Figure D.2. The Effect of the γ hyperparameter on Importance sampling performance.

RCE from 25% to 22.5% would show as -10% on this plot. Overall we can see our method is not very sensitive to this choice, but using a reasonable γ does reduce error by around 10%, with more relative difference on larger sample sizes. We use $\gamma = 0.2$ as it performs the best on sample sizes around 200 that we use in our large study.

D.2.2 Sensitivity to β in BRAGg (Uniform Prior)

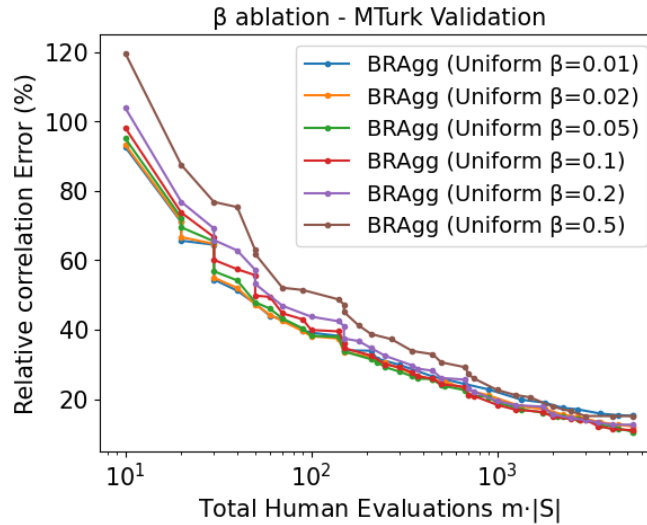


Figure D.3. Comparing the error rate of using different β values for BRAGg(Uniform prior).

Our Rating Aggregation Method 3a Bayes - Uniform Sampling described in Section

4.3.2 depends on the β hyperparameter to select the uniform prior. In this section, we conducted a test on Setting 2 (MTurk test, using MG-IS sampling) comparing different values for β . As shown in Fig D.3, the exact choice of β has little effect on the correlation error, with values between 0.02 and 0.2 performing well. In our experiments we used $\beta = 0.05$ which performs well overall. We think the small sensitivity is mostly because changing prior has a relatively linear effect on predicted c_t , and since correlation coefficient normalizes the c_t the scale of c_t does not change the correlation. Each datapoint in Figure D.3 is the average over 30 samples with different random seeds.

D.2.3 Activation-Guided Importance Sampling

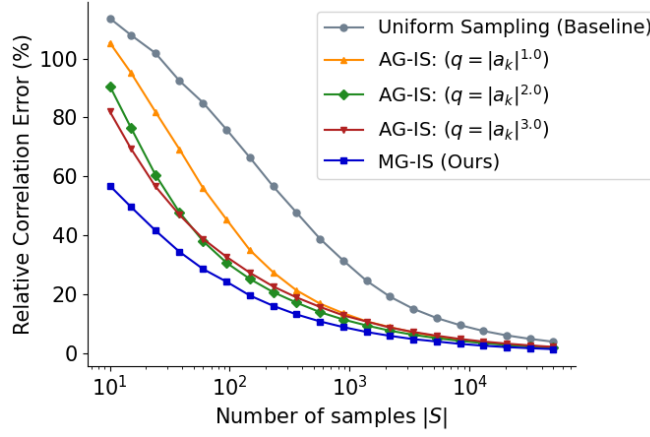


Figure D.4. Comparing different sampling strategies on simulated data with no label noise. We can see that importance sampling without model guidance can still perform quite well and deliver significant gains over uniform sampling.

In addition to Model-Guided Importance Sampling (MG-IS), we also tested importance sampling guided purely by neuron activations, which we call activation guided importance sampling (AG-IS). This could be useful for cases where no cheap model is available to estimate concept presence. In particular we tested with importance sampling functions of the following form:

$$q^{act}(x_i) = \frac{|\tilde{a}_k|_i^\alpha}{\sum_{x_i \in \mathcal{D}} |\tilde{a}_k|_i^\alpha} \quad (\text{D.1})$$

$$q^{\text{AG-IS}}(x) = (1 - \gamma)q^{act}(x) + \gamma p(x) \quad (\text{D.2})$$

As shown in Figure D.4, we can see very significant gains over uniform sampling, with the best performing variant being $\alpha = 2$, but overall performance is not as good as our model-guided importance sampling. We used $\gamma = 0.2$ for these experiments.

D.2.4 Error Model for Rating Aggregation

For our main results, our Bayes modeling uses a uniform error model, i.e. we assume all have concepts and inputs share the same error rate η as described in Section 4.3.2. However, this is not always a realistic assumption, as some inputs and concept are harder/more ambiguous, leading to higher error rates. To account for this, we also experimented using an alternative error model where the error rate η_{ti} is different for each pair of input i and concept t .

In this model $\mathbb{P}(r_{ti}^j = c_{ti}^*) = 1 - \eta_{ti}$, where $\eta_{ti} \sim \text{Beta}(0.75, 2.5)$. The hyperparameters $\alpha = 0.75$ and $\beta = 2.5$ were selected to maximize the probability of the MTurk user ratings we observed in Setting 2.

To do Bayesian inference with this model, we do inference on both c_{ti}^* and η_{ti} and report $[c_t]_i = \int_{\eta_{ti}} \mathbb{P}(C_{ti}, \eta_{ti} | R_{ti}) d\eta_{ti}$.

Applying the Bayes rule $\mathbb{P}(C_{ti}, \eta_{ti} | R_{ti}) = \mathbb{P}(R_{ti} | C_{ti}, \eta_{ti}) \mathbb{P}(C_{ti}, \eta_{ti}) / \mathbb{P}(R_{ti})$ we get:

$$[c_t]_i = \frac{\int_{\eta_{ti}} \mathbb{P}(R_{ti} | C_{ti}, \eta_{ti}) \cdot \mathbb{P}(C_{ti}) \mathbb{P}(\eta_{ti}) d\eta_{ti}}{\int_{\eta_{ti}} \mathbb{P}(R_{ti} | C_{ti}, \eta_{ti}) \cdot \mathbb{P}(C_{ti}) \mathbb{P}(\eta_{ti}) + \mathbb{P}(R_{ti} | \neg C_{ti}, \eta_{ti}) \mathbb{P}(\neg C_{ti}) \mathbb{P}(\eta_{ti}) d\eta_{ti}} \quad (\text{D.3})$$

Since $\mathbb{P}(\eta_{ti})$ and $\mathbb{P}(C_{ti})$ are independent. In practice calculating exact solution for the integral is difficult, so we instead approximate it with 100 discrete bins.

Likelihood: $\mathbb{P}(R_{ti} | C_{ti}, \eta_{ti})$.

With $\alpha_{ti} = \sum_{j=1}^m r_{ti}^j$, we obtain the likelihood in below equations:

$$\mathbb{P}(R_{ti} | C_{ti}, \eta_{ti}) = (1 - \eta_{ti})^{\alpha_{ti}} (\eta_{ti})^{(m - \alpha_{ti})} \quad (\text{D.4})$$

$$\mathbb{P}(R_{ti} | \neg C_{ti}, \eta_{ti}) = (\eta_{ti})^{\alpha_{ti}} (1 - \eta_{ti})^{(m - \alpha_{ti})} \quad (\text{D.5})$$

Priors: For $\mathbb{P}(C_{ti})$ we use the same priors as in Section 4.3.2 (uniform or SigLIP prior).

For $\mathbb{P}(\eta_{ti})$, we use the Beta distribution:

$$\mathbb{P}(\eta_{ti}) \propto \eta_{ti}^{\alpha} \cdot (1 - \eta_{ti})^{\beta} \quad (\text{D.6})$$

The hyperparameters $\alpha = 0.75$ and $\beta = 2.5$ were selected to maximize the probability of the MTurk user ratings we observed.

Results

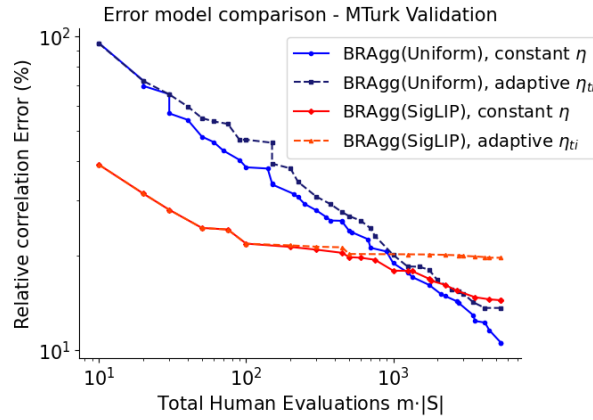


Figure D.5. Comparing different error models for rating aggregation. Note log-scale on both axes.

As shown in Figure D.5, the performance between different error models is similar, but overall the simplified uniform error model (solid lines) leads to slightly better results, so we use the simple error model for our main results. We are not fully certain why the simple model performs better, but one possibility is that even 9 ratings per user is not enough for the Bayes model to accurately infer per-input error rates.

D.2.5 SigLIP Only Evaluation Accuracy

Figure D.6 shows comparison of different rating aggregation strategies vs a pure automated evaluation only relying on SigLIP scores. We can see that the optimal rating strategy depends on your budget. A fully automated evaluation provides similar accuracy as a hybrid evaluation (BRAGg(SigLIP)) with 100 samples per (neuron, explanation) pair, or a pure human

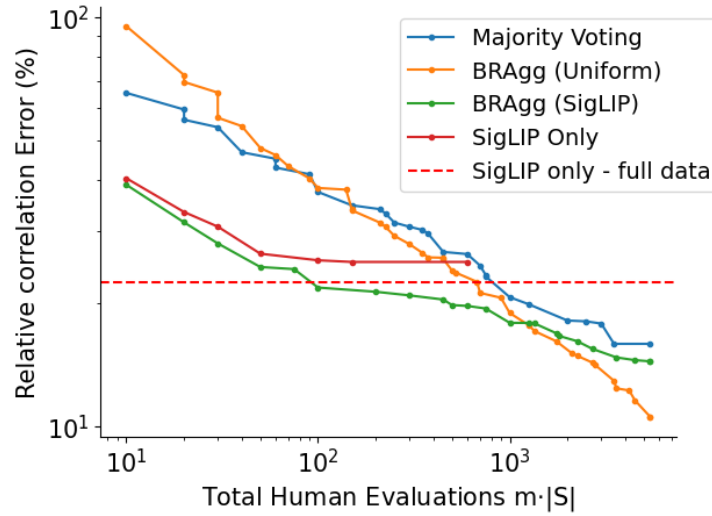


Figure D.6. Comparison the error of different rating aggregation strategies and only relying on SigLIP instead of conducting a human study. Using our Setting 2 results from MTurk with MG-IS sampling.

evaluation with 800 samples per neuron. This implies that the optimal study strategy depends on your budget: If you cannot afford more than 100 human evaluations per neuron, it is best to rely on automated evaluation. For budgets between 100-1000 samples per neuron, a hybrid evaluation (BRAGg(SigLIP)) performs the best. For sufficiently large budget, i.e. > 1000 evaluations per neuron, pure human evaluation (BRAGg(Uniform)) performs the best.

D.2.6 Compositional/Complex Explanations

A more complex explanation is typically more accurate, but it is harder to understand, and more expensive to evaluate as it requires labels for each of the concepts involved in the explanation. We use the length l to indicate explanation complexity, where l is the number of unique concepts in the explanation, e.g. "cat" OR "dog" would have length 2.

For fairness we split the explanations into two groups, simple explanations where $l = 1$ and complex explanations with $l > 1$. Table 4.1 in main text shows comparison of different simple explanations, while Table D.1 compares complex explanation methods. For complex explanations, we can see Linear Explanations reaches the highest correlations with most methods

Table D.1. SigLIP based simulation with correlation scoring comparing complex explanations ($l > 1$). We can see Linear Explanation (LE) overall performs the best.

Complex Exp. ($l, 1$)	Comp Exp [65] $l=3$	INVERT [16] $l=3$	CCE [54], $l=5$	CCE [54], $l=15$	LE(label) [69] $l=4.37/1.97$	LE(SigLIP) [69] $l=4.66/1.82$
RN-50 (Layer4)	0.1399 ± 0.002	0.2341 ± 0.002	0.0993 ± 0.002	0.1510 ± 0.003	<u>0.2924</u> ± 0.002	0.3772 ± 0.002
ViT-B-16 (Layer11 MLP)	0.0468 ± 0.002	0.1101 ± 0.003	0.0534 ± 0.003	0.0570 ± 0.006	<u>0.3243</u> ± 0.005	0.3489 ± 0.005

having higher scores than their simple counterpart, highlighting the value of added complexity.

D.2.7 Model Ablation

In our main results we used the SigLIP [111] ViT-L-16-384 model both as the guide for importance sampling (MG-IS), and to provide the prior for our Bayes rating aggregation. For the Automated Evaluation described in Section 4.5.1 we used the SigLIP ViT-SO400M-14-384 model.

The purpose of this section is to analyze the sensitivity of our results to the model choice. In particular, we test replacing the SigLIP ViT-L-16-384 model for sampling and rating aggregation with the following models:

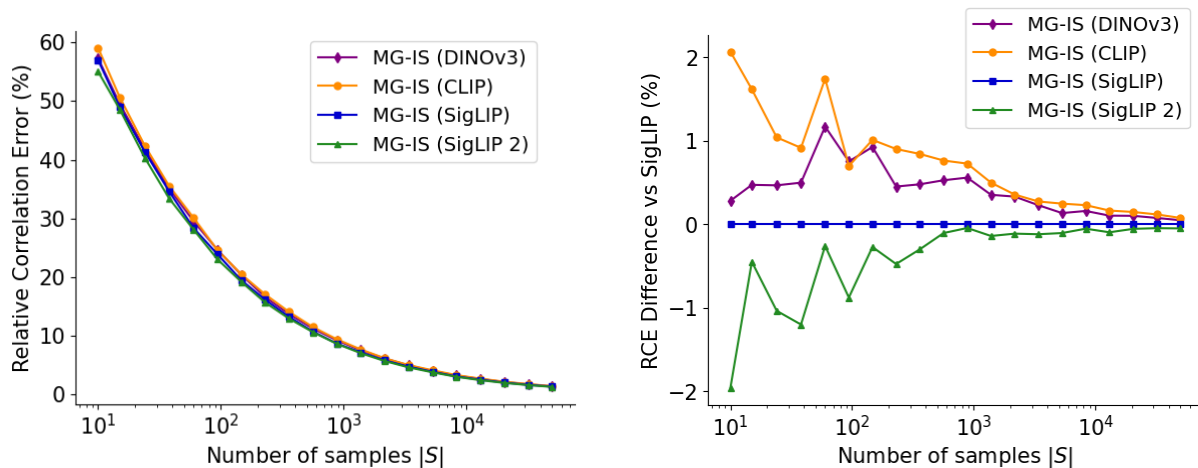
1. CLIP [77] ViT-L14-336, which is an older model with slightly weaker benchmark performance.
2. SigLIP2 [97] ViT-gopt-16-384, which is a bigger and more recent model with slightly better benchmark scores.
3. DINOv3 [87] ViT-L-16 with pretrained head for zero-shot classification.

(*dinov3_vitl16_dinotxt_tet1280d20h24l*)

For all models we used the weights and implementation from the open_clip package [43].

Results - Sampling Model:

Figure D.7 shows a comparison of different models for guiding sampling. We can see better models improve results but the differences are quite small. On average using CLIP instead of SigLIP leads to around 5 % (relative) increase in error with the same number of samples, while using SigLIP 2 leads to around 2 % (relative) reduction in error. DINOv3 performs worse than SigLIP but better than original CLIP on this task.



(a) Comparing different models for guiding sampling in our simulated test (Sett. 1). Stronger models improve sampling but the differences are small.

(b) Same plot, but zooming in on the difference in RCE using SigLIP vs. other models reveals the relative ranking.

Figure D.7. Model ablation for guided sampling. (a) All models converge similarly; (b) the per-model difference from SigLIP shows the relative ranking. Note much smaller y-axis range in (b).

Results - Model for Bayes Prior:

When changing the model used for creating the prior in BRAGg, we observe similar trends, with larger models generally performing better than smaller ones, but the results are a bit less clear and depend on the number of evaluations. Figure D.8 shows the results of using different models as the Bayes prior. We notice a somewhat bigger performance difference in this setting, with for SigLIP 2 reducing RCE at (3 raters per input and 200 inputs) from 19.70% (of SigLIP) to 17.95% for around 10% relative reduction in error. Interestingly, using CLIP as the prior performs worse with small sample sizes but beats SigLIP with large evaluation budget.

Table D.2. Large scale crowdsourced study results using SigLIP 2 as the prior for BRagg.

Model	Aggregation	CLIP-Dissect	INVERT l=1	DnD	MAIA	LE(SigLIP) l=1
ResNet-50	BRagg	0.1392	0.1490	0.1242	0.1072	0.1786
(Layer4)	(SigLIP 2 Prior)	± 0.008	± 0.005	± 0.008	± 0.008	± 0.009
ViT-B-16	BRagg	<u>0.1714</u>	0.1250	0.1039	0.1355	0.2591
(Layer11 mlp)	(SigLIP 2 Prior)	± 0.019	± 0.021	± 0.016	± 0.019	± 0.023
Average	BRagg	<u>0.1553</u>	0.1370	0.1141	0.1201	0.2189
	(SigLIP 2 Prior)					

Inspired by these results, we also analyze our main experiment results using SigLIP 2 instead of SigLIP as the Bayes prior and report the results in Table D.2. Overall we can see the results are very similar to our results using SigLIP as the prior reported in Table D.5. The only difference we notice is the correlation scores are slightly higher for all methods on ViT-B-16 neurons. This highlights our conclusions regarding the relative performance of different explanation methods are not sensitive to model choice.

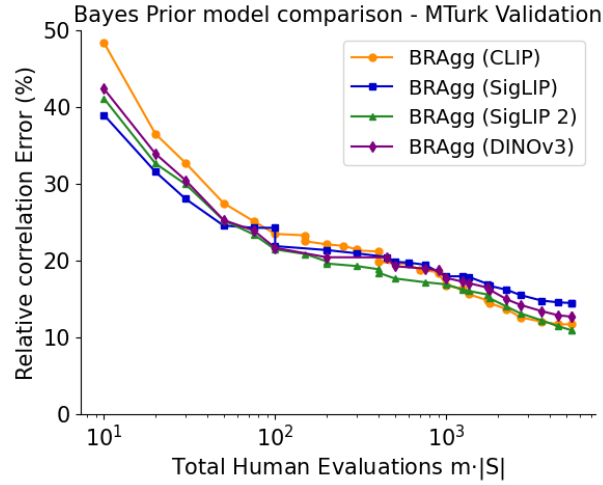
**Figure D.8.** Comparing different choices for the Bayes prior. We can see SigLIP2 performs the best overall.

Table D.3. Comparing Sampling performance based on SigLIP AUC in detecting the concept. We report average RCE.

AUC Range: (#neurons)	0-0.6 (217)	0.6-0.8 (209)	0.8-0.95 (204)	0.95-0.99 (359)	0.99-1 (1059)
Uniform	0.271	0.408	0.447	0.513	0.677
MG-IS	0.268	0.351	0.299	0.210	0.089

D.2.8 Effect of Sampling Model Accuracy on MG-IS Performance

To study how the performance of our sampling model (SigLIP) affects the efficiency of MG-IS, we repeated the experiment in Sec 4.3.1 and split neurons into groups based on how accurately SigLIP can detect their concepts. Table D.3 shows the RCE of different neuron groups when sampling 180 inputs using different methods. As we can see, even on neurons where SigLIP accuracy is close to random chance ($\text{AUC} < 0.6$), our MG-IS still produces slightly lower error than pure uniform sampling baseline. This is likely because MG-IS samples from neurons with a high product $\hat{a}_k \cdot \hat{c}_t$, so even when $\hat{c}_t$ is inaccurate, oversampling high activations can still reduce variance. For concepts that are both rare and low AUC, a larger sample size is needed to estimate correlation with low variance. However as seen in Table D.3, MG-IS still outperforms the uniform sampling baseline in those cases. Conversely, on neurons where SigLIP accuracy is high (most neurons) MG-IS reduces error $\sim 7\times$.

D.2.9 Comparison with Downstream Task Evaluation

To test whether our correlation evaluations give signal on downstream task performance of neuron explanation methods, we ran the evaluation introduced by [4](Appx. A.6), where they measure whether neuron explanation methods can be used to find neurons that make good classifiers for unseen classes. In particular, we tested whether we can find neurons in ResNet-50 layer4 (trained on ImageNet) that make good classifier for CIFAR or Places365 classes. For each class, we find the neuron whose explanation has the highest text embedding similarity to the class name, and then measure it’s performance as a classifier on the new dataset using AUC.

Table D.4. Using generated explanations to find neurons in RN-50 that work as classifiers for unseen classes. We report Average AUC.

Target Classes	CD	DnD	INVERT	LE
CIFAR10	0.763	0.697	0.751	0.787
CIFAR100	0.727	0.723	0.733	0.746
Places365	0.726	0.692	0.679	0.727

We report the average AUC across classes in Table D.4. The intuition is that a good explanation method should be able to identify better neurons for each task. Overall we see that the results of this experiment align with our results and the methods that scored highest in our correlation evaluation ($LE > CD$, $INVERT > DnD$) also find the best classifier neurons.

Study information

► Click to View Study Information

☐ By checking this box I indicate that I am at least 18 years old, have read the study information above, and agree to participate in this research study.

Task

Select all the images that contain: **ground beetle**.

If you do not know what **ground beetle** means, use a tool like Google Image search to find out.



Figure D.9. Our user study interface.

D.3 Additional Details

D.3.1 MTurk Experimental Details:

Figure D.9 showcases the full user interface displayed to the raters. We selected raters based in the US with over 10,000 tasks approved and $> 98\%$ task approval rate. Each rater was paid \$0.05 per task (and we paid another \$0.01 per task in fees to MTurk), which takes around 15 seconds to complete based on our testing, for an estimated \$12/hr earnings.

Table D.5. Detailed results of our large scale crowdsourced study. For each model we tested 100 random neurons, and we report the average correlation score of the explanations produced by each method, as well as the standard error of the mean. Comparing different rating aggregation methods, we can see that the correlation scores themselves vary a lot depending on the aggregation method used, but the order between explanations methods is quite consistent.

Model	Aggregation	CD[68]	INVERT l=1[16]	DnD[4]	MAIA[86]	LE(SigLIP) l=1[69]
ResNet-50 (Layer4)	Majority	0.0978 ± 0.010	<u>0.0981</u> ± 0.010	0.0978 ± 0.010	0.0834 ± 0.010	0.1216 ± 0.011
	BRAgg (Uniform Prior)	<u>0.1017</u> ± 0.009	0.0977 ± 0.010	0.1011 ± 0.008	0.0876 ± 0.010	0.1261 ± 0.011
	BRAgg (SigLIP Prior)	0.1356 ± 0.008	<u>0.1472</u> ± 0.005	0.1179 ± 0.007	0.0992 ± 0.008	0.1803 ± 0.008
ViT-B-16 (Layer11 mlp)	Majority vote	<u>0.1035</u> ± 0.016	0.0754 ± 0.016	0.0825 ± 0.017	0.0723 ± 0.013	0.1208 ± 0.015
	BRAgg (Uniform Prior)	<u>0.0934</u> ± 0.014	0.0637 ± 0.014	0.0689 ± 0.013	0.0555 ± 0.012	0.1018 ± 0.012
	BRAgg (SigLIP Prior)	<u>0.1568</u> ± 0.017	0.1167 ± 0.020	0.0904 ± 0.015	0.1257 ± 0.017	0.2450 ± 0.021
Average	BRAgg (SigLIP Prior)	<u>0.1462</u>	0.132	0.1042	0.1125	0.2127
	All	<u>0.1148</u>	0.0998	0.0931	0.0873	0.1493

D.3.2 Detailed MTurk results

Table D.5 shows more detailed results of our crowdsourced evaluation, including multiple rating aggregation methods. Comparing different rating aggregation methods, we can see that the correlation scores themselves vary a lot depending on the aggregation method used, but the order between explanations methods is quite consistent, for example Linear Explanations [69] score the highest regardless of aggregation method used.

Statistical Significance

Table D.6 reports the statistical significance of our results using the two-sample Student’s t-test. We can see we have statistically significant evidence that Linear Explanations performs better than all the other methods, with the exception of CLIP-Dissect when using BRAgg with uniform prior, and all methods with the SigLIP prior.

Table D.6. Statistical significance of our results according to two-sample students t-test. We report the p-values of different hypothesis, with statistically significant p-values in bold.

Model	Aggregation	Hypothesis			
		LE >CD	LE >INVERT	LE >DnD	LE >MAIA
RN-50	BRAgg(Uniform)	0.0601	0.0079	0.0198	0.0010
	BRAgg(SigLIP)	0.0020	0.0098	0.0000	0.0000
ViT-B-16	BRAgg(Uniform)	0.4295	0.0037	0.0431	0.0062
	BRAgg(SigLIP)	0.0039	0.0001	0.0000	0.0001

Example Ratings

Figures D.10, D.11 and D.12 showcase example neurons and the descriptions assigned by different explanation methods, as well as the correlation scores estimated by our crowd-sourced study for those explanations. We colored explanations based on the estimated correlation coefficient, with Green: $\rho > 0.25$, Yellow: $0.25 \geq \rho > 0.10$ and Red: $0.10 \geq \rho$.

We can see sometimes generative methods like MAIA [86] produce the best explanations, for example "Vibrant Green Elements in Nature" in Figure D.10(Neuron 1126), but other times are too specific or fail to find the right concept D.11.

Overall we did not observe very high correlation scores for any neurons in ResNet50, likely due to them being more densely activated and polysemantic, see for example neuron 108 (Fig. D.10 that activates for both trains and invertebrates. On the other hand, for ViT-B-16 we observed several neurons that were extremely interpretable (correlation scores ≥ 0.5), such as Neuron 2187 (Fig. D.12) that seems to only activate on Alpine ibex/Mountain goats.

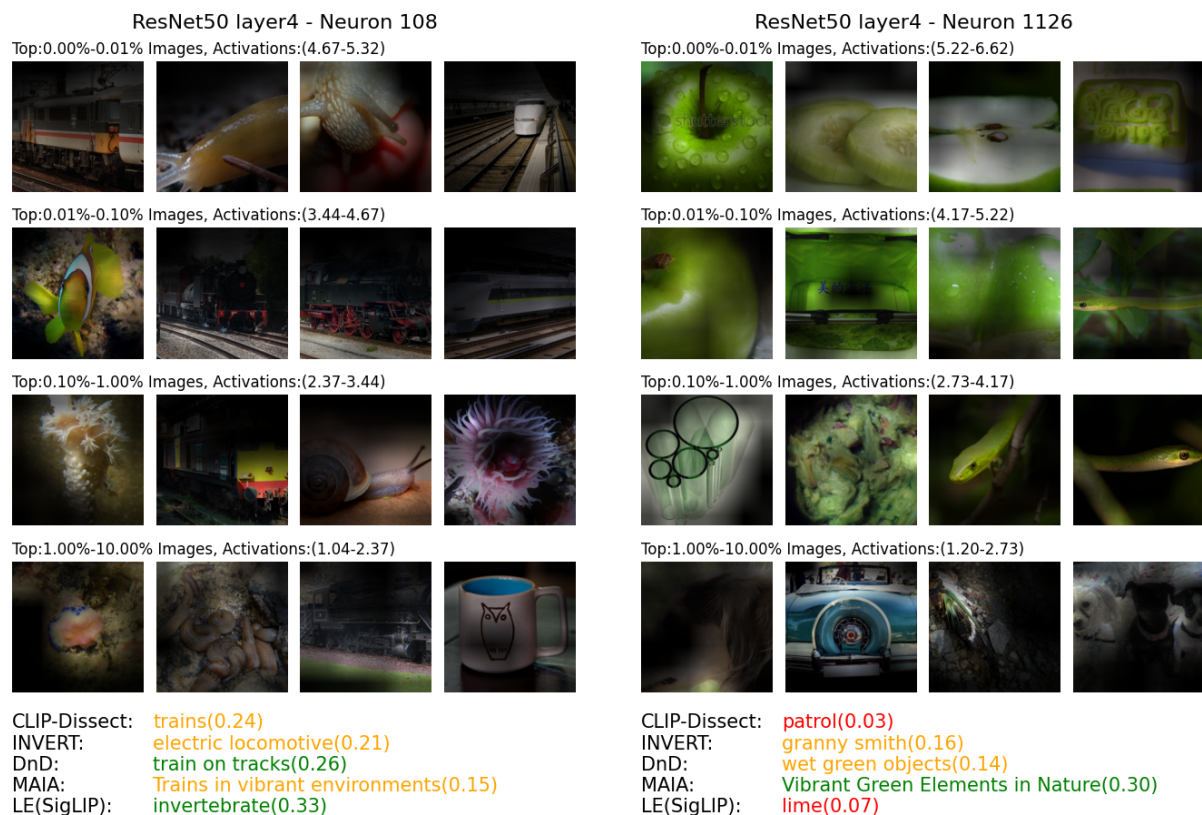


Figure D.10. Visualization of example neurons, their descriptions and correlations scores from our crowdsourced evaluation (BRagg with SigLIP prior). We have colored the descriptions based on the correlation score.

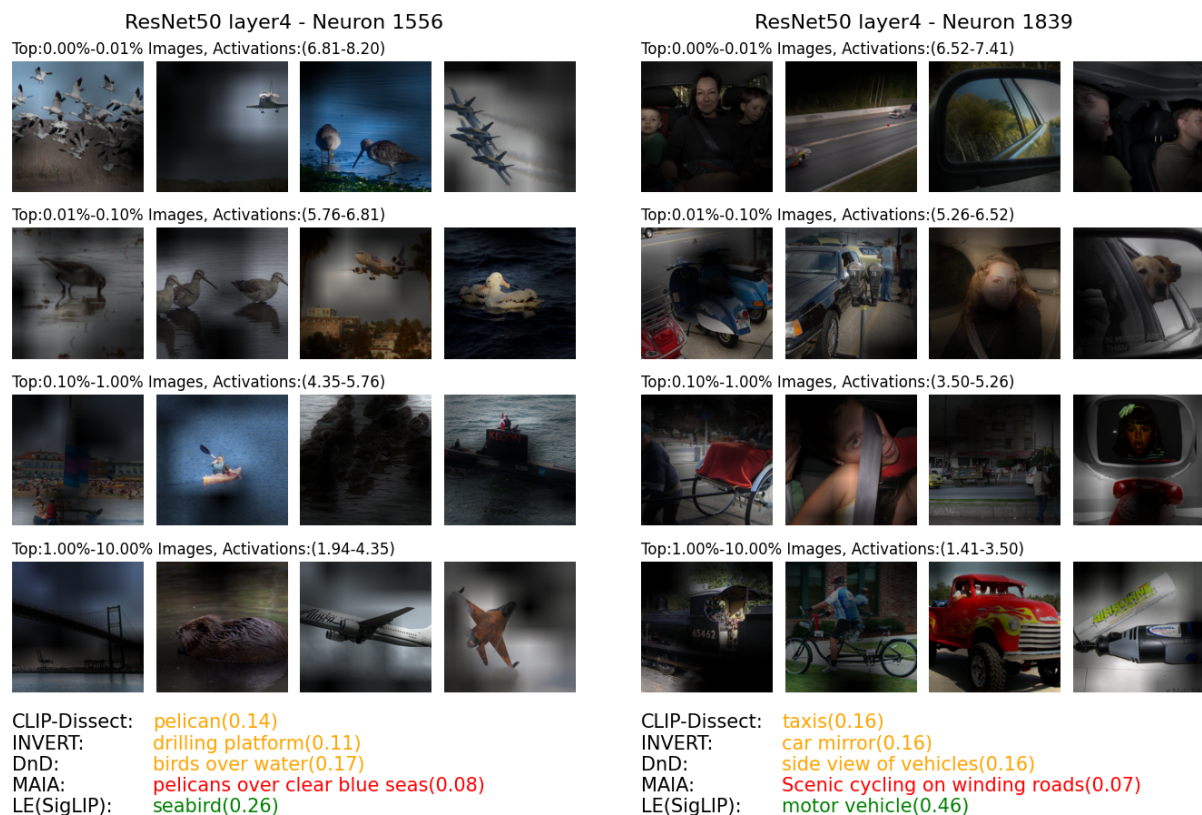


Figure D.11. Visualization of example neurons, their descriptions and correlations scores from our crowdsourced evaluation (BRagg with SigLIP prior). We have colored the descriptions based on the correlation score.



Figure D.12. Visualization of example neurons, their descriptions and correlations scores from our crowdsourced evaluation (BRagg with SigLIP prior). We have colored the descriptions based on the correlation score.

D.3.3 Experimental Details

Our evaluation focused on two models/layers:

1. Layer4 (end of residual block 4) of ResNet-50 trained on ImageNet. For simulation we use the neuron activation after average pooling, giving scalar activations, but for methods designed for 2d-activations, activations before average pooling are given as input.
2. MLP-activations in Layer 11 of ViT-B-16 [20] Encoder trained on ImageNet. We focus only on the activations of the CLS-token as this is the last layer and other tokens do not affect the prediction.

We use the full ImageNet validation dataset as $\mathcal{D}$ for the human study for all methods, and for generating explanations unless the method requires a specific dataset for explanation generation such as Broden [6]. For all methods using SigLIP guidance we use the SigLIP-SO400M-14-386 model.

Baseline Implementation Details

For practical purposes we made slight modifications to some baseline methods. The changes are discussed in detail below:

DnD [4]: The original implementation uses GPT-3.5 Turbo through the OpenAI API. Given the high cost of using the API and that recent open-source LLMs have strong performance compared to even closed-source LLMs like GPT-3.5, we replace GPT-3.5 Turbo with Llama 3.1-8B-Instruct [33]. DnD [4] showed that the older Llama 2 model was already better than GPT-3.5 Turbo and on-par with GPT-4 Turbo for neuron description, so our choice of Llama 3.1 does not degrade DnD’s performance w.r.t. GPT-3.5 Turbo.

MAIA [86]: Compared to the original implementation, we replace GPT4-vision-preview with the newer GPT4o-2024-08-06 [74] since it has lower API costs and better performance. The method is still quite expensive, costing us approximately \$65 and \$116 to generate descriptions for 100 randomly selected neurons of ResNet-50 Layer4 and ViT-B-16 Layer11 MLP respectively.

Note that this cost also includes repeating the experiment for ~ 10 and ~ 20 neurons from ResNet and ViT respectively which did not yield any neuron description in the first run. Initially, we also tried open-source LLMs with support for vision inputs (*i.e.* VLMs) like Llama-3.2-11B-Vision-Instruct [33], Llava-OneVision-Qwen2-7B-ov-hf [56], and CogVLM2-Llama3-Chat-19B [40]. However, these VLMs do not work well with MAIA since they fail to generate executable code, forget the image tokens and focus only on the last few text tokens if given a long prompt, and allow only one image input at a time. This is likely because these VLMs are geared towards visual question answering and do not possess the more generalized capabilities of GPT4/4o.

Methods relying on 2d activations: Many methods are designed to explain entire channels of CNNs with 2d activations [6, 65, 39, 54]. For ResNet-50 layer4 we fed the pre-avg pool activations to these methods for proper 2d input. However, for ViT-B-16 last layer only the CLS-token activations affect the output, and as such we are explaining neurons with scalar activations. In this case, we broadcasted the scalar activations into a 2d-tensor with the same value in all spatial locations. However this is not the intended way to use these methods and may partially explain poorer performance of some methods on ViT-neurons as observed in Tables 4.1 and D.1.

CCE [54]: For Clustered Compositional Explanations, we tested two different versions: the $l = 15$ version corresponds to the default version with length 3 explanations for each of the 5 activation clusters. For the $l = 5$ version we used explanation length=1 with 5 clusters of activations. We also used the implementation of [54] to reproduce the results of Compositional Explanations [65] by setting explanation length=3 and number of clusters=1.

Subset of Neurons

For most methods in Table 4.1 and D.1, we report the average correlation scores across all 2048 neurons for ResNet-50 layer4 and 3072 neurons for ViT-B-16 layer11 mlp. However for certain methods due to high computational and/or API cost we were only able to explain a subset of these neurons and report the average score of these subsets in Tables 4.1 and D.1. We

report the results for a subset of neurons for the following methods:

- MAIA [86]: Randomly selected subset of 100 neurons each for both RN50 and ViT-B-16.
- CCE [54] $l = 5$: For ViT-B-16 we used a subset of 1420 neurons. RN50 evaluated on all neurons.
- CCE [54] $l = 15$: RN50: subset of 984 neurons. ViT-B-16: subset of 422 neurons.

All other methods were evaluated on all neurons of each layer.

D.3.4 Automated Evaluation Details

For our automated evaluation (Sec. 4.5.1), we use Simulation with Correlation Scoring as described by [69]. This evaluation was originally proposed for language model neurons by [9].

The basic idea of simulation evaluation is to use the *explanation* to predict neuron activations on unseen inputs. With correlation scoring we then evaluate the correlation coefficient ρ between the predicted activations s and actual neuron activations a_k on the entire test split of 10,000 inputs as done by [69].

For **simple explanations**, the predicted activation s is simply the presence of concept on that input.

$$s(x_i, t) = [c_t]_i \quad (\text{D.7})$$

For a **linear explanation**, $E = \{(w_1, t_1), \dots, (w_l, t_l)\}$ the predicted activation s is calculated following [69] as:

$$s(x_i, E) = \sum_{w_j, t_j \in E} w_j [c_{t_j}]_i \quad (\text{D.8})$$

For **compositional explanations** [65], we calculate the predicted activation as follows using probabilistic logic. Different basic logical operators are calculated as:

$$s(x_i, t_1 \text{ AND } t_2) = [c_{t_1}]_i \cdot [c_{t_2}]_i \quad (\text{D.9})$$

$$s(x_i, t_1 \text{ OR } t_2) = 1 - (1 - [c_{t_1}]_i) \cdot (1 - [c_{t_2}]_i) \quad (\text{D.10})$$

$$s(x_i, \text{NOT } t) = 1 - [c_t]_i \quad (\text{D.11})$$

Predictions for more complex compositions are then calculated by iteratively applying these rules.

Clustered Compositional Explanations: CCE [54] explanations are of the form $E = \{(l_1, u_1, F_1), \dots, (l_r, u_r, F_r)\}$ where r is the number of activation clusters, and l_j, u_j are the lower and upper bound of activations for that cluster and F_j is a compositional explanation for activations of that cluster. To predict neuron activation based on this explanation, we use the following formula:

$$s(x_i, E) = \sum_{l_j, u_j, F_j \in E} \frac{l_j + u_j}{2} s(x_i, F_j) \quad (\text{D.12})$$

This means if the concepts according to the formula of a cluster are present, we predict the neuron's activation will be in the middle of the clusters activation range.

For all automated evaluations we use SigLIP-SO400M-14-386 model to predict c_t following [69].

D.3.5 Theorem 1

Suppose we are estimating the expected value of function $h(x)$ when $x \sim \mathcal{P}$. Let $\mathcal{X}$ be the support of $\mathcal{P}$.

Theorem 1 ([81], Sec 3.3.2, Theorem 3.12). *For importance sampling with sampling distribution*

q :

$$\mathbb{E}_{x \sim \mathcal{P}}[h(x)] = \int_{\mathcal{X}} h(x) \frac{p(x)}{q(x)} q(x) dx \approx \frac{1}{|S|} \sum_{i=1}^{|S|} \frac{h(x_i) p(x_i)}{q(x_i)}.$$

The choice of q that minimizes the variance satisfies $q(x) \propto |h(x)|p(x)$.

Proof: Reproduced from [81].

$$\text{Var} \left[\frac{h(x)p(x)}{q(x)} \right] = \mathbb{E}_q \left[\left(\frac{h(x)p(x)}{q(x)} \right)^2 \right] - \left(\mathbb{E}_q \left[\frac{h(x)p(x)}{q(x)} \right] \right)^2 \quad (\text{D.13})$$

Since the the second term $\left(\mathbb{E}_q \left[\frac{h(x)p(x)}{q(x)} \right] \right)^2 = \left(\int_{\mathcal{X}} h(x)p(x) dx \right)^2$ does not depend on q , in order to minimize variance we only need to minimize the first term.

From Jensen's inequality it follows that:

$$\mathbb{E}_q \left[\left(\frac{h(x)p(x)}{q(x)} \right)^2 \right] \geq \left(\mathbb{E}_q \left[\frac{|h(x)|p(x)}{q(x)} \right] \right)^2 = \left(\int_{\mathcal{X}} |h(x)|p(x) dx \right)^2 \quad (\text{D.14})$$

Giving us a lower bound for the first term. If we set

$$q(x) = \frac{|h(x)|p(x)}{\int_{\mathcal{X}} |h(z)|p(z) dz} \quad (\text{D.15})$$

Which is a valid probability distribution, we get:

$$\mathbb{E}_q \left[\left(\frac{h(x)p(x)}{q(x)} \right)^2 \right] = \left(\int_{\mathcal{X}} |h(z)|p(z) dz \right)^2 \quad (\text{D.16})$$

This exactly matches the lower bound, proving that minimum variance is attained by setting

$$q(x) = \frac{|h(x)|p(x)}{\int_{\mathcal{X}} |h(z)|p(z) dz} \propto |h(x)|p(x) \quad (\text{D.17})$$

D.3.6 Compute details

Our main contribution is focused on efficient crowd-sourced evaluation and as such our method is not computationally costly. The main computational cost associated with our method is calculating the SigLIP image encoder outputs for the entire $\mathcal{D}$, as these are needed for both importance sampling and the Bayes - SigLIP prior. However this is a relatively cheap onetime cost of around 20 minutes on a single NVIDIA RTX 6000 Ada Generation GPU.

The main computational expense associated with this paper involved running the baseline methods. In Tables D.7 and D.8 below, we report the approximate runtime to explain all neurons of a layer using different description methods using a single NVIDIA RTX 6000 Ada Generation GPU.

Table D.7. Approximate runtime of different $l = 1$ baseline methods to explain neurons.

Simple Exp. $l = 1$	ND [7]	MILAN [39]	CD [68]	INVERT $l=1$ [16]	DnD [4]	MAIA [86]	LE(label) $l=1$ [69]	LE(SigLIP) $l=1$ [69]
RN-50 (Layer4)	~ 1 hr	~ 1 hr	~ 5 mins	~ 1 hr	~ 55 hrs	~ 255 hrs	~ 1 hr	~ 1 hr

Table D.8. Approximate runtime of different complex explanation baseline methods.

Complex Exp. $l > 1$	Comp Exp [65] $l=3$	INVERT [16] $l=3$	CCE [54], $l=5$	CCE [54], $l=15$	LE(label) [69] $l=4.37/1.97$	LE(SigLIP) [69] $l=4.66/1.82$
RN-50 (Layer4)	~ 92 hrs	~ 24 hrs	~ 87 hrs	~ 275 hrs	~ 1 hr	~ 1 hr

D.3.7 Inter-Annotator Consistency

To measure the quality of our ratings, we also measure inter-annotator consistency through Fleiss’s Kappa. This measures the amount of agreement above chance, with 1 meaning perfect agreement and 0 random chance.

For MTurk raters in our test Setting 2, we observe a Kappa of 0.395. On the real study we observe Fleiss’s Kappa of 0.191 on ViT neuron explanations and 0.274 on RN50 neurons. Overall

we can see these agreement rates are relatively low, highlighting the need for error correction through rating aggregation. While we think some of the noise in responses is underlying poor quality responses, some disagreement is also caused by underlying ambiguity in the task or poor explanations like MAIA sometimes outputting "Undetermined Selectivity".

To compare against higher quality annotations, we had 3 authors rate 150 samples each for the 10 neurons in Setting 2. For authors we observed a Fleiss Kappa of 0.774. This shows that obtaining higher quality raters has potential to improve study signal, but as we show we can still efficiently utilize noisy ratings with our methods.

Appendix E

Supplementary Material for Chapter 5

E.1 Additional Details and Results

E.1.1 Appendix Overview

In this section we provide a brief overview of the Appendix contents. The Appendix is mostly focused on Ablation studies identifying the importance and effectiveness of different components of our pipeline, as well additional Figures to showcase more examples of our results.

First in section E.1.2 we discuss the limitations of our method, then in section E.1.3 we show that our proposed *cos cubed* similarity function performs well in describing neuron functionality. In section E.1.4 we perform an ablation study on the effect of different concept set filters we have proposed, and in Sec E.1.5 we discuss our reasoning behind each filter in detail through an example. In section E.1.6 we show that our GPT-3 generated concept set helps us reach higher accuracy than previous methods for creating initial concept set. In Appendix E.1.7 we show the effects of removing the sparsity constraint on our models, leading to higher accuracy but reduced interpretability. In Appendix E.1.8 we show our results are consistent despite inherent randomness in our pipeline. Next, in Appendix E.1.9 we provide a detailed explanation of the procedure we used for model editing described in Sec 5.5.2. Finally in Sections E.1.10, E.1.11 and E.1.12 we provide additional figures for model editing, individual decision explanations and model weight visualizations respectively. In Appendix E.2 we provide results of our large-scale crowdsourced human evaluation, which confirms that LF-CBM is indeed more

interpretable than standard models according to two metrics.

E.1.2 Limitations

While our model performs well in terms of both interpretability and accuracy, it still has some limitations. For example, using a model like GPT-3 to produce concepts is stochastic, and sometimes may fail to generate important concepts for detecting a certain class. However our model is quite robust to small changes in the concept set as shown in sections E.1.4 and E.1.8. In general, automatic ways of generating concept set may sometimes lack the required domain knowledge to include some important concepts, and would perhaps be best used in collaboration with a human expert.

Second, our model is aimed to extend Concept Bottleneck Models to large datasets where labeled concept data is not available. However it works best on domains where CLIP performs well and as such may not perform as well on small datasets that require specific domain knowledge and labeled concept data is available, such as medical datasets. For such datasets we believe it is still better to use a method that can effectively leverage the labels available such as original Concept Bottleneck Models [50] or P-CBM[108].

E.1.3 Dissection performance of *cos cubed* similarity function

In this section we discuss how good the *cos cubed* similarity function we defined in section 5.3.2 is at describing neuron functionality, i.e. how well it performs as a similarity function for CLIP-Dissect. Specifically we measured its performance on describing the roles of final layer neurons on a ResNet-50 trained on ImageNet (where we know the ground truth role of the neurons) in terms of both accuracy and average distance from the correct description in a text embedding space. We followed the quantitative evaluation methodology of [68], which describes this process in more detail. We compared the results against using CLIP-Dissect with two different similarity functions, *softWPMI* which was the best function identified by [68] and *cos* which is a simple differentiable baseline. We can see *cos cubed* performs as well as

softWPMI in terms of similarity in text embedding space, while being slightly worse in terms of accuracy, and is much better than simple *cos* similarity. This is impressive given *cos cubed* is fully differentiable while *SoftWPMI* is not differentiable.

Table E.1. Comparison of the performance between similarity functions. We look at the final layer of ResNet-50 trained on ImageNet. We use 20,000 most common English words as the concept set for mpnet cos similarity and ImageNet classes as the concept set for top1 accuracy. We can see *cos cubed* performs much better than simple *cos*, and almost as well as *SoftWPMI*.

Metric	Similarity function	D_{probe}					Differentiable
		CIFAR100 train	Broden	ImageNet val	ImageNet val + Broden	Average	
mpnet	cos	0.2761	0.215	0.2823	0.2584	0.2580	Yes
cos similarity	SoftWPMI	0.3664	0.3945	0.5257	0.5233	0.4525	No
	cos cubed	0.3579	0.4101	0.5187	0.5223	0.4523	Yes
Top1 accuracy	cos	8.50%	5.70%	15.9%	11.4%	10.38 %	Yes
	SoftWPMI	46.20%	70.50 %	95.00%	95.40%	76.78%	No
	cos cubed	31.00%	49.40 %	87.40%	85.50%	63.33%	Yes

E.1.4 Ablation: Effect of concept filters

In this section we study how each step in our proposed concept filtering effects the results of our method. In general our use of filters has two main aims:

- To improve the interpretability of our models
- Improve computational efficiency and complexity by reducing the number of concepts

The filters are not designed to improve performance of the models in terms of accuracy, in fact we would expect them to slightly reduce accuracy as a model with more concepts is generally larger and more powerful.

To evaluate the effect each individual filter has we trained models on both CIFAR10 and ImageNet while deactivating one filter at a time, as well as one without using any filters at all. The results are displayed in Table E.2. We can see that the accuracy of our models is not at all sensitive to the choice of filters, with ImageNet accuracy remaining constant and CIFAR10 slightly increasing its accuracy with less filters. We were unable to train an ImageNet model without any filters as training with the large number of concepts required more memory than we had available in our system. Additionally the ImageNet model without similarity to other concepts filter had worse accuracy and less sparsity than other tested models, which we think may be caused by the GLM-SAGA optimizer not finding as good of a solution when the number of concepts increased too much.

Table E.2. Effect of our individual concept filters on the final accuracy and number of concepts used by our models. *We were unable to train the model to be sparse enough, results from a less sparse model.

Filters	CIFAR10 Accuracy	CIFAR10 #concepts	ImageNet Accuracy	ImageNet #concepts
All filters	86.26%	142	71.89%	4380
No length filter	86.55%	146	71.88%	5347
No similarity to classes filter	86.74%	148	71.93%	4671
No similarity to other concepts filter	86.42%	151	70.90%*	6515
No CLIP activation filter	86.41%	147	71.92%	4478
No projection accuracy filter	86.33%	147	71.88%	4462
No filters at all	86.56%	177	N/A	9087

E.1.5 Further discussion and example of concept filtering

To further show the effects of each filter, we will showcase the full filtering procedure for our CIFAR10 model with all filters. We start with a freshly generated initial concept set from GPT-3, which has a total of 177 concepts. We then take the following steps to filter down concepts:

1. **Delete concepts that are too long.** This step is important to make sure concepts are easy to visualize and simple non-convoluted concepts. For CIFAR-10 this leads to deletion of 3 concepts:

- *white spots on the fur (in some cases), feline features (e.g., whiskers, ears) and legs with two toes pointing forward and two toes pointing backward.*

174 concepts remain.

2. **Delete concepts too similar to output classes.** This step is required for the explanations to be informative, as it helps avoid trivial explanations such as: "This is image is a cat because it is a cat". This step removes the following CIFAR-10 concepts (with the class name it is too close to shown in brackets):

- a cat (cat), a deer (deer), a horse (horse), a plane (airplane), a car (automobile), car

(automobile), cars (automobile), vehicle (automobile), animal (dog), truck driver (truck)

164 concepts remain

3. **Delete concepts too similar to other concepts.** In this step we aim to delete duplicate concepts from the concept set, i.e. two concepts that have the same semantic meaning should not both be part of the concept set. Duplicate concepts will cause unnecessary computational cost and confusing explanations. For CIFAR-10 we delete the following concepts (with the concept it is too similar to in brackets):

- 4 wheels (four wheels), a food bowl (a bowl), a furry, four-legged animal (a large, four-legged mammal), a gasoline station (a gas station), a large boxy body (a large body), a large, muscular body (a large body), a street (a road), a strong engine (an engine), large, bulging eyes (large eyes), the ocean (the sea)

154 concepts remain

4. **Remove concepts not present in training data.** In this step we remove all concepts that CLIP thinks are not really present in training data. The purpose is to avoid learning neurons that don't correctly represent their target concepts, as concepts missing from the dataset are unlikely to be learned correctly in the Concept Bottleneck Layer. For CIFAR-10 this step deletes the following concepts:

- a bit, a mechanic, legs, long legs, passengers, several seats inside, windows all around

147 concepts remain

5. **Remove concepts we can't project accurately.** The purpose of this step is similar to the previous step, we want to remove all concepts that are not faithfully represented by the CBL. To do this we evaluate the similarity score between the target concept and the activations of our new neuron using CLIP-Dissect [68] on the validation data, and delete concepts with similarity less than 0.45 which was determined to be a good indicator of faithful concept representation. For CIFAR-10 we delete the following concepts:

- a bed, a coffee mug, a crew, a house, a wheel

We are left with our 142 final concepts.

In total our method has 5 different cutoff parameters that can be tuned. Since the main purpose of these filters is to make the network more interpretable, their values were mainly chosen through trial and error to find ones that produce the most useful and explainable models. Most of these cutoffs are independent of the dataset chosen, and the only one we changed from one dataset to another was the value of cutoff for images being present in the dataset (filter 4). We used a cutoff of 0.25 for CIFAR-10 and CIFAR-100, 0.26 for CUB-200 and 0.28 for Places and ImageNet. We found we had to change this cutoff as CLIP doesn't activate very highly on low resolution images of CIFAR, and in general we get higher top5 activations on larger datasets. Since the other 4 cutoffs are fixed, using our method for a new dataset won't require a large hyperparameter search, and as seen in Table E.2 the overall performance is not very sensitive to specific filters.

To visualize the effect of our filters, in Figure E.1 we visualize the final layer weights for CIFAR-10 class "automobile" trained with and without filters. We can see the model with filters has found a reasonable decision rule, with largest weights corresponding to *four wheels* and *a steering wheel*. On the other hand, the weights for the model without filters are quite problematic. First, the 3 largest weights are *cars*, *car* and *a car*. Not only do they have the same meaning as the final class itself (filter 2), thus lacking any explanatory power, they are duplicates of each other (filter 3), making the explanation unnecessarily complicated. Finally we see the unrelated concept *bed* which is not present in CIFAR-10 having a small positive weight, which should be removed by either filter 4 or 5.

E.1.6 Ablation: Effect of Initial Concept set

In this section we evaluate the effect our initial concept set generator has on our results. In our paper we proposed a new method of generating descriptions using GPT-3 [15], and in this section we compare those results to the case where we generate the initial concept set using

CIFAR-10 final layer weight visualization

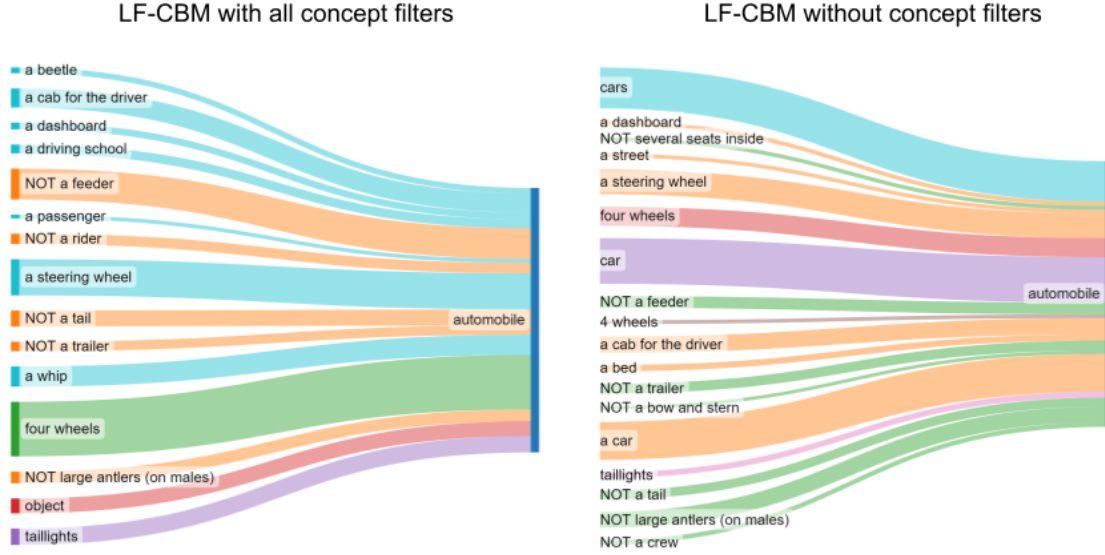


Figure E.1. Comparison of the final layer weights between CIFAR-10 model trained with and without filters.

ConceptNet [90] as proposed by [108]. In general ConceptNet only works well when prompted with relatively common single word phrases, but many of class names for CUB200, Places365 and ImageNet consist of multiple words, such as the ImageNet class *great white shark*. To overcome this issue we split multi-word class names into single word components and add concepts related to each word to the concept set, in this case *great*, *white* and *shark*. After creating the initial concept set we use the same filtering procedure described in section 5.3.1B for both models. Table E.3 displays the accuracy comparison between models trained on different initial concept sets. In general using GPT3 seems to provide a small (0.1-1.5%) accuracy boost on all datasets, with the exception of CUB200 where the ConceptNet model fails completely. This is because ConceptNet is unable to generate concepts for the highly specific class names in CUB200, such as *Groove billed Ani*, while GPT-3 is not troubled by this.

Table E.3. Accuracy comparison between using different methods for generating initial concept set.

Model\Dataset	CIFAR10	CIFAR100	CUB200	Places365	ImageNet
LF-CBM (GPT-3 [original])	86.40%	65.13%	74.31%	43.68%	71.95%
LF-CBM (ConceptNet)	86.30%	63.62%	2.19%	42.49%	71.52%

E.1.7 Ablation: LF-CBM without sparsity constraint

To estimate the effect sparsity has on our models, we show the accuracy of LF-CBM models trained without sparsity constraints (LF-CBM (dense)) in Table E.4. We can see that not using a sparsity constraint greatly improves accuracy (except for CUB200 which starts to overfit), but it comes at a very large cost for interpretability as can be seen in Figure E.2, where almost none of the decision is explained by the 10 most highly contributing concepts.

Table E.4. Results of training our CBM without sparsity constraints on the final layer.

Model\Dataset	CIFAR10	CIFAR100	CUB200	Places365	ImageNet
LF-CBM (sparse [original])	86.40%	65.13%	74.31%	43.68%	71.95%
LF-CBM (dense)	87.50%	67.93%	74.25%	48.25%	74.09%

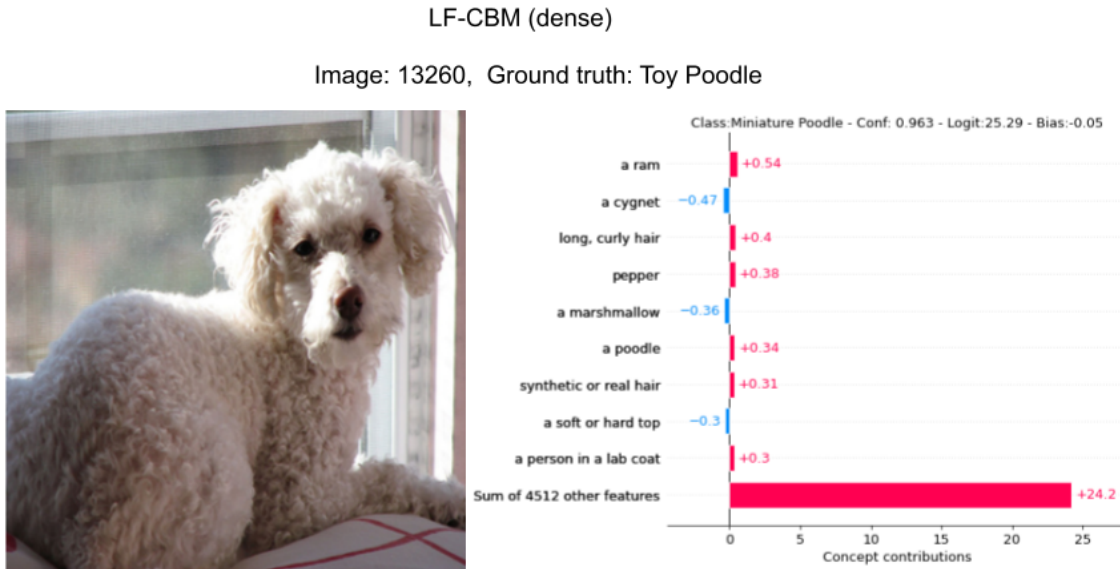


Figure E.2. Dense CBM explanation for the same image as the fourth image in Figure E.11. We can see the dense model is practically uninterpretable.

E.1.8 Consistency of concept set generation

Since the initial concept set is generated using GPT-3, it is a random process and in this section we explore how much this noise effects our final results. When regenerating initial concept set and running our concept filtering pipeline for our ImageNet model, we got 4380 concepts, compared to the original 4523 concepts. In terms of accuracy, the model learned with regenerated concept set reached an accuracy of 71.89%, compared to the average of 71.95% with the original concept set. This indicates that the difference is very small in terms of accuracy. However, the exact concepts used and model weights can be pretty different as shown in Figure E.3, despite all being relevant (e.g. there are many concepts that appear on both concept sets, e.g. Image 49962 has important concepts of "wipes", "toiletries" on both, though they have different corresponding contributions). Figure E.3 shows the explanations for decisions on two random images for the original model and model trained with regenerated concept set.

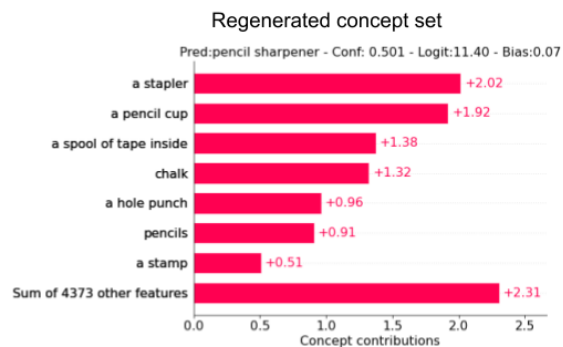
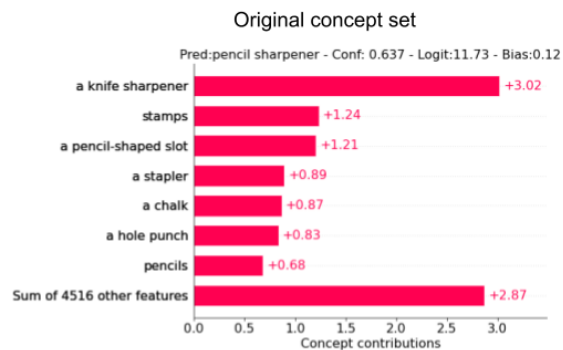
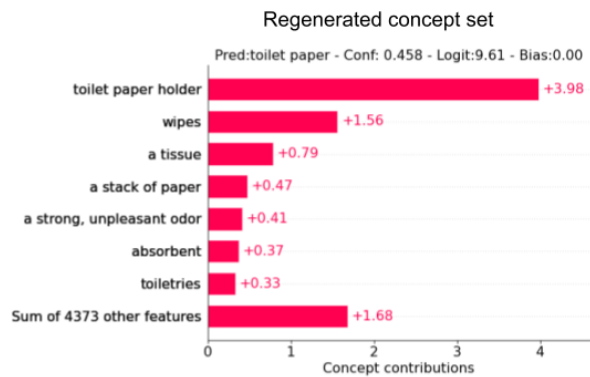
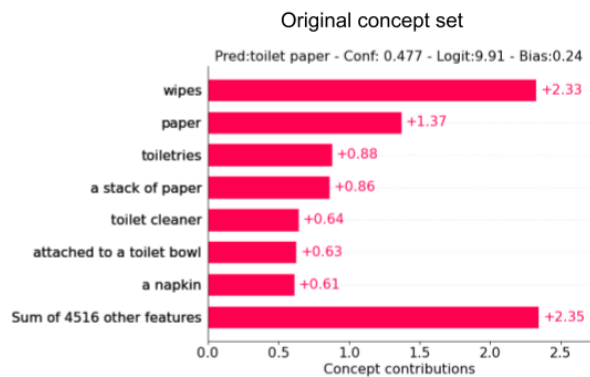


Figure E.3. Difference in concepts used for a decision and their weights for models trained with original vs regenerated concept set, shown on two random images.

E.1.9 More details on Model editing procedure

Here we provide an improved explanation of Sec 5.5.2 on how we can edit the final layer weights of our proposed Label-Free CBM to correct incorrect predictions. Our procedure includes 3 steps:

Step 1: Find an input image x_i where the model makes a Type 4 Error (incorrect final layer weight):

This is done by visualizing incorrect model predictions and their explanations similar to Figure 5.4. To identify if it is a Type 4 error, first we check that the ground truth label of the input image is correct and unambiguous (therefore not type 1 error described in Sec 5.5.1). Next, we check that the highly activating concepts look correct (therefore not type 2 or type 3 error described in Sec 5.5.1).

Step 2: Identify a concept to edit:

We start by listing the highest activating concepts for the chosen input, i.e. the highest elements of $f_c(x_i)$ (after normalization). This allows us to identify concepts that are the most important for this specific image while not too important for other images. From these concepts we use our domain knowledge (and/or internet search) to understand which concepts are relevant to making this decision.

For example, the image in the right panel of Figure 5.6 is originally predicted as a “Shopping basket”, while the ground truth is a “Hamper”. The 5 most highly activating concepts for this image are: “a basket”, “made of rope or string”, “a rope”, “a laundry basket” and “a fishing net”. After a short investigation we find that although the classes “Hamper” and “Shopping basket” are very similar, hampers are more often constructed in a woven like manner. Therefore, we identify the concept of “made of rope or string” as relevant for capturing this difference, and choose it for editing in the next step.

Step 3: Change weights by sufficient magnitude:

Once we have selected the concept to be corrected, we will compute how much the

magnitude of the associated final layer weight should be changed (denoted as Δw , $\Delta w \in \mathbb{R}$) and then edit the weights in the following way:

$$W_{F[gt,concept]} \leftarrow W_{F[gt,concept]} + \Delta w, \quad W_{F[pred,concept]} \leftarrow W_{F[pred,concept]} - \Delta w \quad (\text{E.1})$$

Note that $W_{F[i,j]}$ denotes the (i, j) element in the matrix W_F . The goal of editing is to correct the inaccurate prediction on this specific instance while minimizing effect on other predictions. To calculate Δw needed to flip prediction, we first calculate the difference in logits Δa (before softmax) before the edit, $\Delta a = W_{F[pred,:]}f_c(x_i) - W_{F[gt,:]}f_c(x_i)$.

Since we would like the prediction to be corrected to gt class, our goal is to design Δw such that

$$(W_{F[gt,:]} + \Delta w \cdot e)f_c(x_i) - (W_{F[pred,:]} - \Delta w \cdot e)f_c(x_i) > 0 \quad (\text{E.2})$$

where e is a one-hot row vector with the entry $e_{[concept]} = 1$. Let $b = (W_{F[gt,:]} + \Delta w \cdot e)f_c(x_i) - (W_{F[pred,:]} - \Delta w \cdot e)f_c(x_i)$, where b is a nonnegative constant deciding how large of a margin we want the correct prediction to have. Since $\Delta a = W_{F[pred,:]}f_c(x_i) - W_{F[gt,:]}f_c(x_i)$, we have:

$$2\Delta w \cdot f_c(x_i)_{[concept]} - \Delta a = b \Rightarrow \Delta w = (\Delta a + b) / (2f_c(x_i)_{[concept]}) \quad (\text{E.3})$$

Typically we use values for b between 0.2 and 2 to calculate required Δw .

It is worth noting that when the model weights are edited, it might affect other image's predictions too due to the change of weight parameters. Thus, with the goal to correct the wrong predictions while not affecting other already correct predictions, we suggest each edit should be checked with a validation dataset before applying on a target model. As described in the end of Sec 5.5.2 in the manuscript, with the above proposed model weight editing, we are able to improve the overall model accuracy from 71.98% to 72.02% by performing the model editing for only 5 different images, which is a non-negligible improvement. Since the edits only affect weights for 10/1000 classes, this corresponds to a 4% accuracy boost on the affected classes.

E.1.10 Additional figures

Figures E.4, E.5 provide examples of the full prompts we used for GPT-3, as well as GPT outputs. For all experiments we used the text-davinci-002 model available through OpenAI API.

Figure E.6 shows the additional model edits performed in our ImageNet CBM experiment, and Figure E.7 showcases Type 1 and Type 2 errors made by our ImageNet CBM.

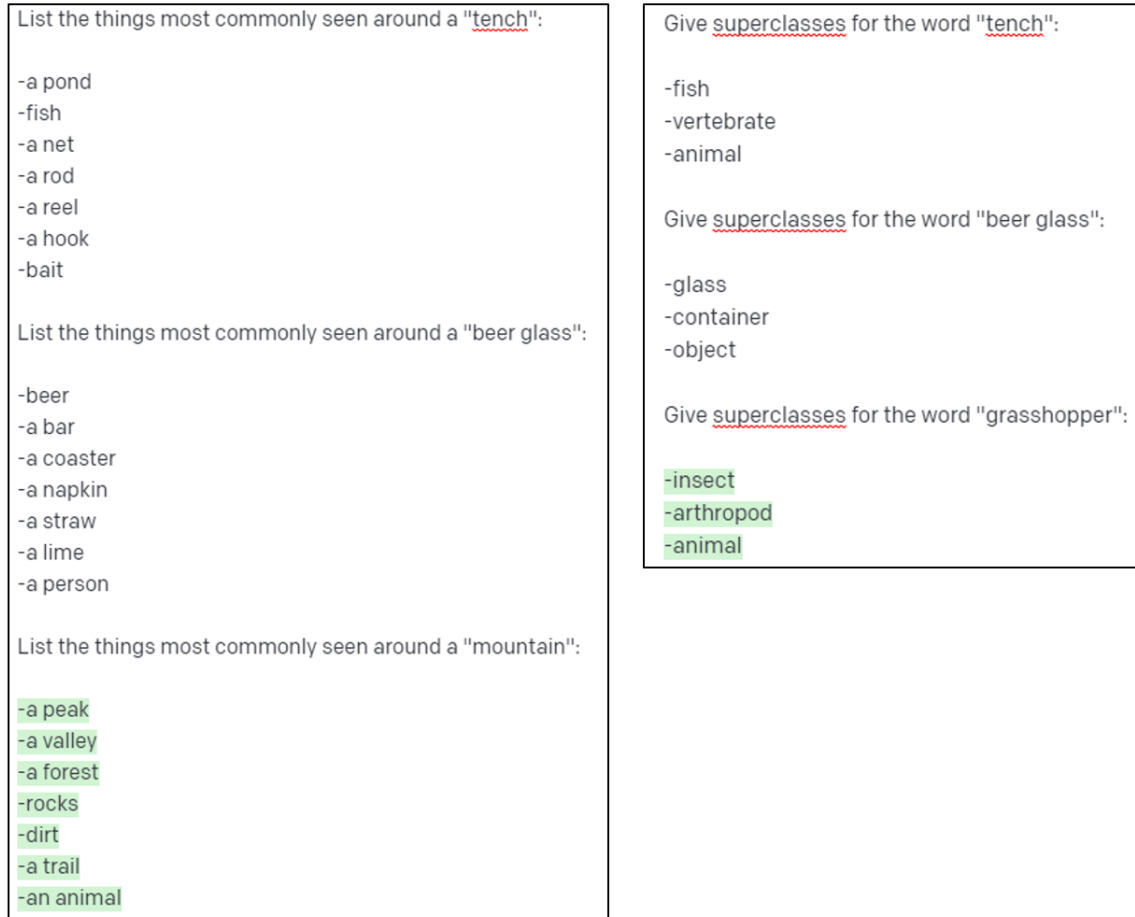


Figure E.4. Full prompts used for GPT-3 concept set creation. Text in green generated by GPT, rest is our prompt.

List the most important features for recognizing something as a "goldfish":

- bright orange color
- a small, round body
- a long, flowing tail
- a small mouth
- orange fins

List the most important features for recognizing something as a "beerglass":

- a tall, cylindrical shape
- clear or translucent color
- opening at the top
- a sturdy base
- a handle

List the most important features for recognizing something as a "scuba diver":

- a person wearing a scuba mask and breathing apparatus
- a wet suit
- flippers
- a diving tank

Figure E.5. Full prompts used for GPT-3 concept set creation. Text in green generated by GPT, rest is our prompt.

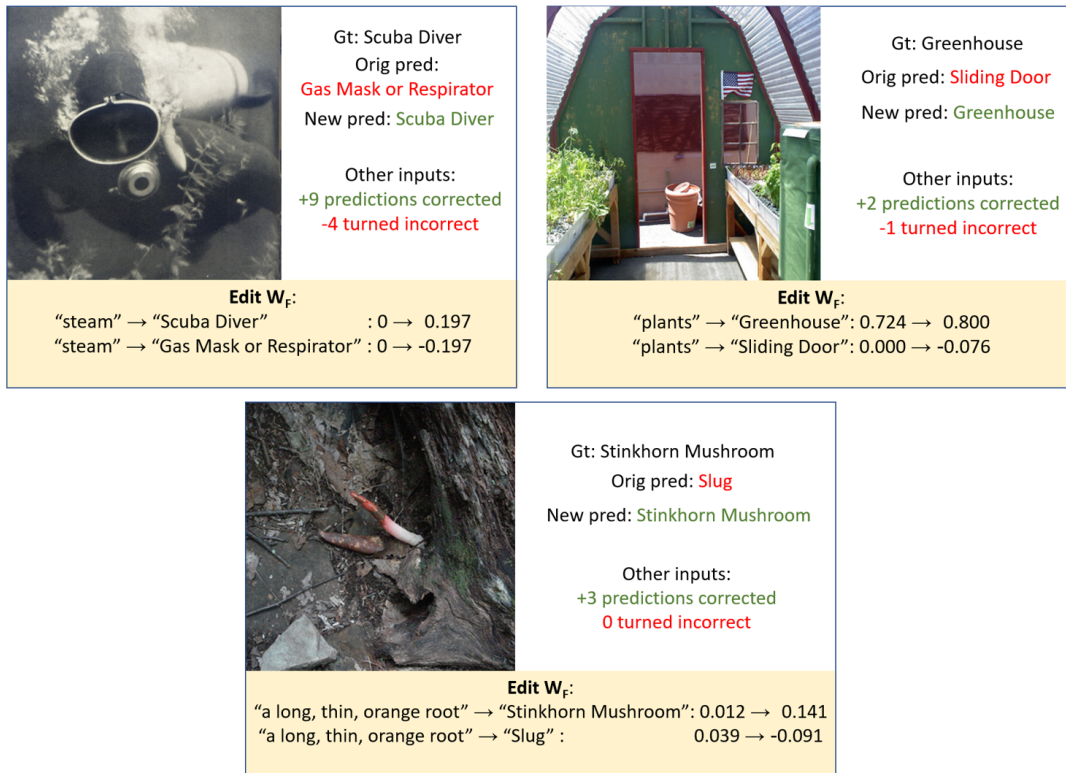


Figure E.6. The other 3 model edits we performed in our experiment

Type 1

Incorrect label

Gt: patas monkey, pred: lion



Ambiguous label

Gt: stethoscope, pred: maltese (dog)



Type2: No sufficient concept in CBL

Gt: barn spider, pred: European garden spider

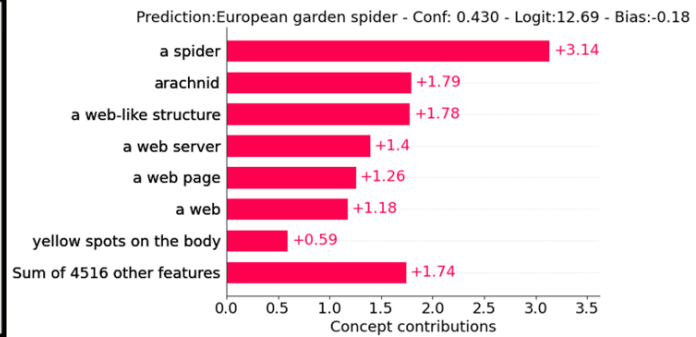


Figure E.7. Examples of Type 1 and type 2 model errors.

E.1.11 Additional weight visualizations for random classes

In figures E.8 and E.9 we have visualized the final layer weights leading to 4 randomly chosen output classes for both Places365 and ImageNet. These were created using the same procedure as Figure 5.3, with minor differences such as focus on single class at a time and no manual editing of concept colors according to semantic similarity. We can see that the concepts that have large weights are indeed relevant to the class, and overall the weights look reasonable.

Places365 - Final layer weight visualizations for random classes



Figure E.8. Weight visualizations for 4 randomly chosen output classes for our CBM trained on Places365.

ImageNet - Final layer weight visualizations for random classes

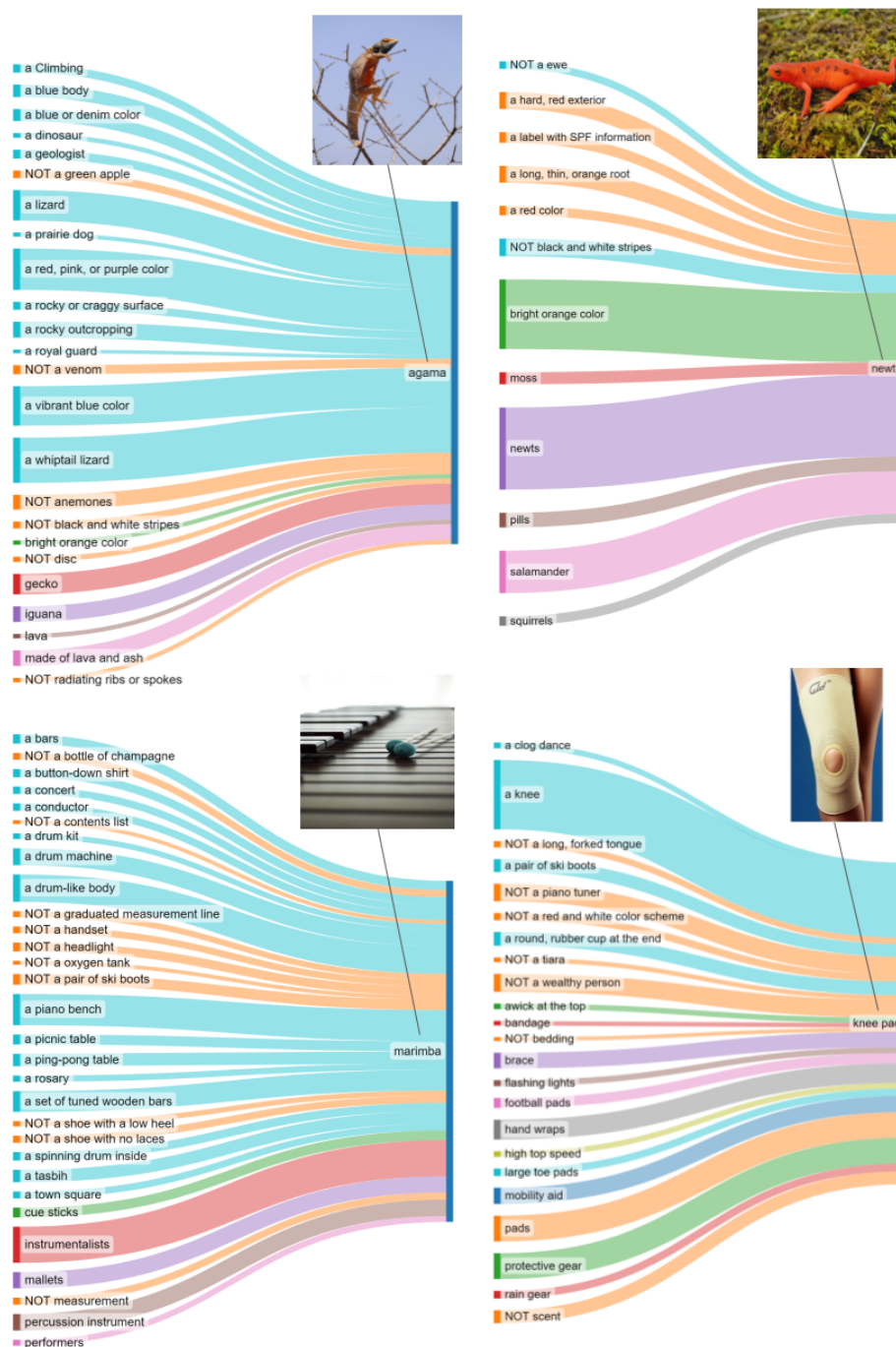


Figure E.9. Weight visualizations for 4 randomly chosen output classes for our CBM trained on ImageNet, as well as an example image of each class to clarify their meaning.

E.1.12 Additional explanations for random images

In figures E.10, E.11, E.12 we display additional samples of decision explanations for 4 randomly chosen input images for each of Places365, ImageNet and CUB-200 respectively. Overall, it can be seen that the concepts and explanations are of good quality, even with the CUB-200 dataset which may require very specific knowledge and concepts to create CBMs. This result demonstrates the effectiveness of our LF-CBM, which does not use the expert-knowledge concept sets unlike existing CBMs.

Places365

Image:19587, Ground truth: Jacuzzi, indoor

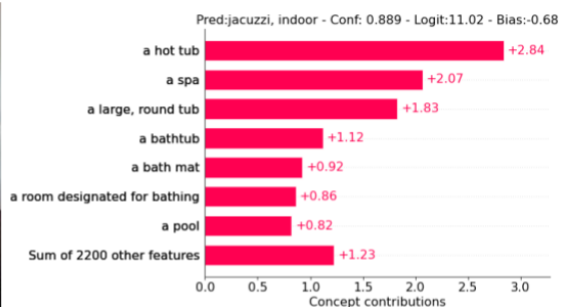


Image:5776, Ground truth: boardwalk

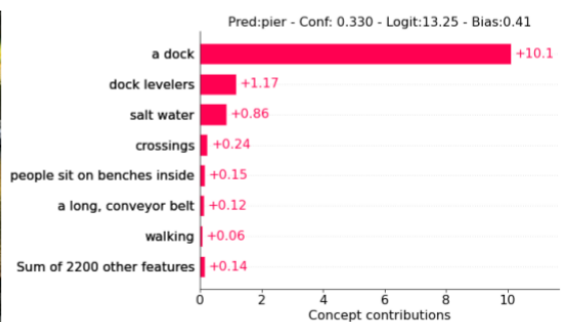
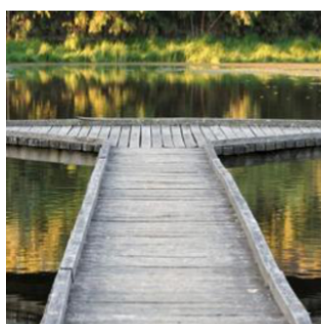


Image: 17442, Ground truth: heliport

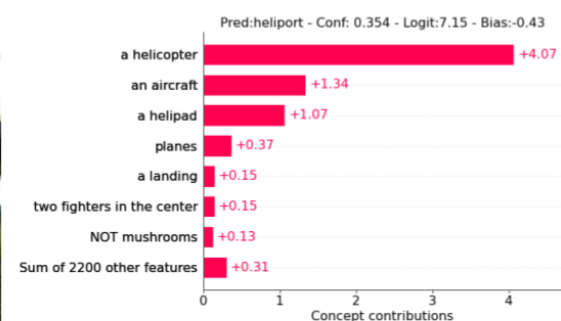


Image: 33310, Ground truth: Topiary Garden

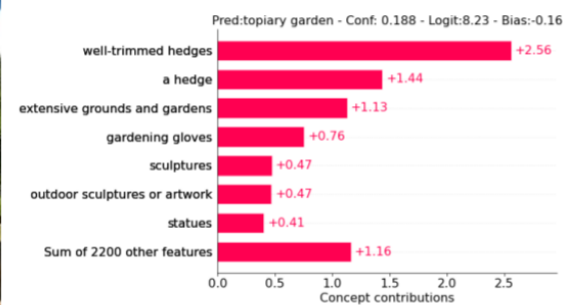


Figure E.10. Explanations for 4 randomly chosen input images for our CBM trained on Places365.

ImageNet

Image:17237, Ground truth: hippopotamus

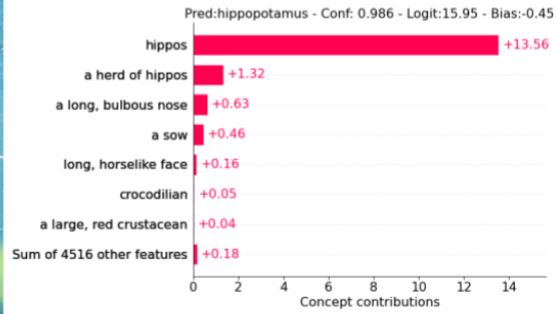


Image:33159, Ground truth: monastery

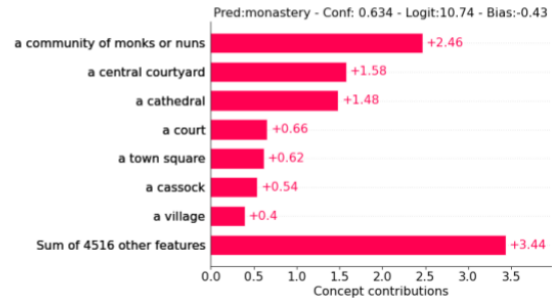


Image: 23074, Ground truth: breastplate

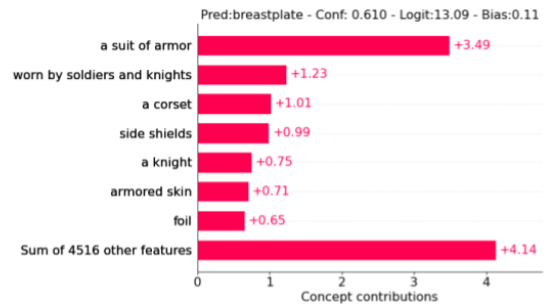


Image: 13260, Ground truth: Toy Poodle

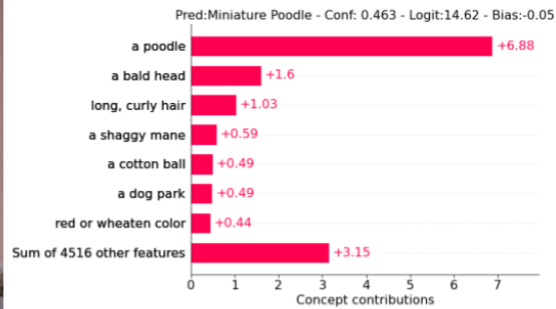
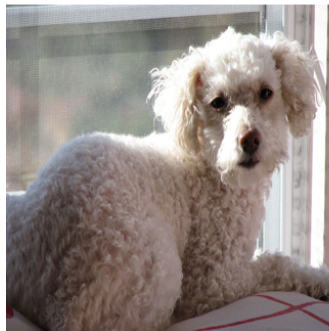


Figure E.11. Explanations for 4 randomly chosen input images for our CBM trained on ImageNet.

CUB200

Image:1169, Ground truth: Yellow bellied Flycatcher

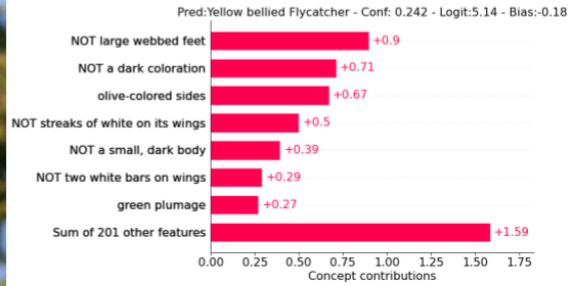


Image:4121, Ground truth: Black Tern

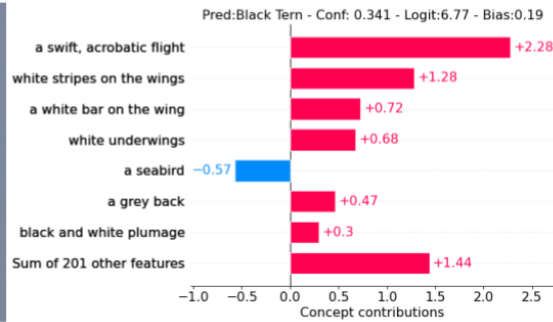


Image: 5159, Ground truth: Tennessee Warbler

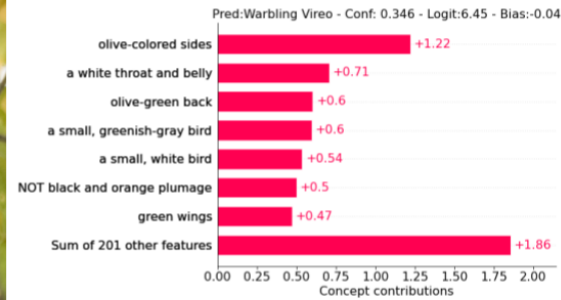


Image: 4823, Ground truth: Kentucky Warbler

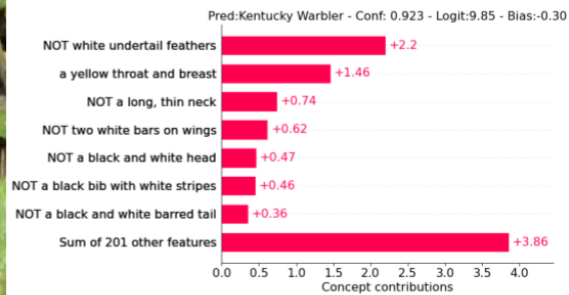


Figure E.12. Explanations for 4 randomly chosen input images for our CBM trained on CUB-200.

E.2 User Study on the Interpretability of LF-CBM

Note: The results in this section were added after the ICLR camera ready deadline to further support the interpretability of our proposed method and are not included in the official ICLR paper.

To get a quantitative measure of the interpretability of our models, we conducted a large scale study on Amazon Mechanical Turk. We evaluated our models on two tasks:

1. Evaluating how well the neurons in CBL correspond to their target concepts
2. Evaluating how reasonable the explanations for the decisions of our models are

E.2.1 Task 1: Do neurons in CBL correspond to their target concept?

For our explanations to be faithful, it is very important that the neurons we learned in the Concept Bottleneck Layer (Section 5.3.2) indeed correspond to their target concepts.

Setup: To test this, we conducted a large scale evaluation on Amazon Mechanical Turk, where we evaluated all 4505 neurons in the CBL of our trained ImageNet LF-CBM, with 3 ratings per neuron. Our experiment interface is shown in Figure E.13. The basic idea is to show users the 10 most highly activating images (from the validation dataset) for a neuron, and ask them to rate whether the description accurately matches the images on a 5-point scale, where 1=Strongly Disagree and 5=Strongly Agree. The description for a CBL neuron is its target concept.

As a baseline, we compared against neurons in the second to last layer of ResNet-50 trained on ImageNet, which was the backbone model used for ImageNet LF-CBM. Since these neurons don't originally have an associated concept, we used CLIP-Dissect [68] to generate explanations for each neuron, and used those as the description. We evaluated each of the 2048 neurons in the layer with 3 raters each. For CLIP-Dissect we used original SoftWPMI similarity function and ImageNet validation data as probing dataset.

Table E.5. Mechanical Turk results of Task 1. We can see the CBL neurons in LF-CBM were deemed to be significantly more interpretable than neurons in its standard backbone network, on average reaching "Agree" scores. Avg score indicates the average rating across all neurons and all evaluations, reported with standard error of the mean.

	Avg score:	1 - Strongly Disagree	2 - Disagree	3 - Neither Agree or Disagree	4 - Agree	5 - Strongly Agree
LF-CBM	3.91 $\pm$ 0.01	4.3%	9.2%	10.1%	43.6%	32.8%
ResNet-50	3.65 $\pm$ 0.02	6.2%	13.9%	14.7%	39.2%	26.0%

We used raters in the United States, with greater than 98% approval rating and at least 10000 previously approved HITs. Users were paid \$0.05 per task. Our experiment was deemed exempt from IRB approval by the UCSD IRB Board.

Results: The results of this experiment are presented in Table E.5. We can see that on average, the neurons in CBL are more interpretable than the neurons of a standard neural network, reaching average score of 3.91 vs 3.65. Since we evaluated thousands of neurons this result is highly statistically significant. For more intuitive numbers, 76.4% of raters gave CBL neurons a rating of "Agree" or higher, while this number was only 65.2% for standard neurons.

Study information

[Click Here to View](#)

☐ By checking this box I indicate that you I am at least 18 years old, have read the study information above, and agree to participate in this research study.

Task

Description for the images: Police or Sheriff markings



Instructions Shortcuts Do you agree with the statement below?

Description: "Police or Sheriff markings". Does the description accurately describe most of the above 10 images?

Instructions ✕

Rate your agreement with the statement: "description" accurately describes most of the above 10 images. If you don't understand the description we encourage you to find out using for example a dictionary or Google Image search.

Select an option

- Strongly Disagree 1
- Disagree 2
- Neither Agree nor Disagree 3
- Agree 4
- Strongly Agree 5

Figure E.13. User Interface shown to Mechanical Turk users for Task 1.

Finally, to evaluate how reliable these crowdsourced results are, we had 4 authors perform the same experiment in a blind setting, each evaluating 100 neurons for both models. Overall the results were similar, we noticed authors were more critical of both descriptions but both sets of evaluators found LF-CBM neurons more interpretable. The average author score was 3.66 for LF-CBM neurons, and 3.30 for neurons in ResNet-50. MTurk average scores for this subset were LF-CBM:3.93, ResNet-50: 3.68.

E.2.2 Task 2: How reasonable are our explanations for a single decision?

In the second task, we wanted to evaluate whether users consider the explanations our LF-CBM produces for individual decisions to be reasonable.

Setup: We show workers explanations based on which concepts contributed the most to the decision similar to Figure 5.4. However, we simplify this explanation to make it more suitable for crowdsourcing. Instead of displaying barplots and contribution magnitudes, we only show the names of 5 most contributing concepts. In addition, to avoid confusion we only show concepts with positive activation and positive weights. In this task we evaluated pairs of explanations to see how LF-CBM explanations compare with explanations from baseline models. Users were asked to rate which explanation is more reasonable and why. The task interface is shown in Figure E.14. We compared our ImageNet LF-CBM against 3 baselines to ablate which parts of our model are most important for interpretability. The baselines are:

- Standard (dense) - ResNet50 trained on Imagenet
- Standard (sparse) - ResNet50 with sparse final layer
- LF-CBM (dense) - Our LF-CBM but with a dense final layer W_F

To create the explanations from standard models, we identified the neurons with highest contributions to the prediction, and used CLIP-Dissect [68] to get their concept. LF-CBM was compared against each baseline on 350 random input images, with 3 raters for each comparison

Instructions

Evaluate and justify which AI model provides a more reasonable explanation for the prediction it made.

▼ Click to View Detailed Instructions

Detailed Instructions:

The purpose of this task is to evaluate which AI model is more reasonably able to explain its predictions. You are shown an input image and two predictions of what the image contains, where both models make the same prediction but for different reasons.

Each explanation displays the 5 most important features/concepts that the model thinks are present in or related to the current image and which contribute the most to the prediction being made. If you do not understand some word/phrase in the prediction or explanation we encourage you to find out using a dictionary or Google Image search.

We believe a reasonable explanation should have the following 3 properties, but you are free to also use other reasons when making decisions (if you do please explain in Other).

Relevant to image: Features of the explanation should be present in the image.

Relevant to prediction: Features of the explanation should be important to deciding whether the image belongs to the predicted class. For example, if the prediction is "Volcano", the feature "lava" would be relevant while the feature "turtle" would not be relevant.

Informative: The explanation should reveal some information about how the decision was made. For example, if the prediction is "Volcano", the feature "volcano" is not informative because it is the same as the final class we are predicting, but features like "lava" and "crater" would be informative.

Task

Image:



Model 1 predicts this image is a "digital clock"

Explanation:
Because the image has the following features (in order of importance):

1. an alarm
2. a countdown
3. a second hand or digital timer
4. a timer
5. a nightstand

Model 2 predicts this image is a "digital clock"

Explanation:
Because the image has the following features (in order of importance):

1. mask
2. ui
3. manufacturer
4. petrol
5. crossword

Which Explanation is more reasonable?

☐ Model 1 Clearly More Reasonable
 ☐ Model 1 Slightly More Reasonable
 ☐ Both Models Equally Reasonable
 ☐ Model 2 Slightly More Reasonable
 ☐ Model 2 Clearly More Reasonable

Why? Select all that apply

☐ The more reasonable explanation uses features that are more **relevant to the image**.
☐ The more reasonable explanation uses features are more **relevant to the prediction**.
☐ The more reasonable explanation is more **informative**.
☐ N/A, both explanations equally reasonable.

Other: _____

Submit

Figure E.14. User Interface shown to Mechanical Turk users for Task 2. In this example Model 1 is LF-CBM while Model 2 is standard dense model.

for a total of 3150 comparisons. We only used inputs where all 4 models made the correct prediction, so the only difference was the explanations themselves.

We again used workers based in the US, with greater than 98% approval rating and 10,000 HITs completed. Each worker was paid \$0.08 per comparison.

Results:

The results of this experiment are displayed in Figures E.15 and E.16. In Figure E.15 we can see users clearly preferred LF-CBM over Standard (dense) and Standard (sparse), with 64.2% and 65.2%(respectively) of users saying LF-CBM explanations are more reasonable

(either slightly or clearly).

However, we found the responses were quite noisy for this task, and to improve their quality, we filtered away responses from any worker who gave an inconsistent response (for example selecting "Both Models Equally Reasonable" and "The More reasonable explanation is more informative" at the same time). The filtered results are shown in Figure E.16. After filtering, the benefits of LF-CBM are even more clear, with 74.4% of ratings saying LF-CBM explanations are more reasonable than Standard (dense) and 78.8% are more reasonable than Standard (sparse).

Interestingly, we found that sparsity did not really improve interpretability in this evaluation for either LF-CBM or the standard model. This was against our expectations, but we think it might be because this evaluation does not take into account how comprehensive the explanation is. A dense and sparse model often have similar top-5 most highly contributing concepts, but for dense concept the top-5 only explain a few percentage of the decision, while for sparse model they often explain 60-70%, as seen in Figures E.2 and E.11.

As for why users found one or the other explanation more reasonable, the most common reason was "More Relevant to the Image"(selected 80% of the time), followed by "More Relevant to Prediction"(60%), while "More Informative" was the least popular, only selected around 30% of the time.

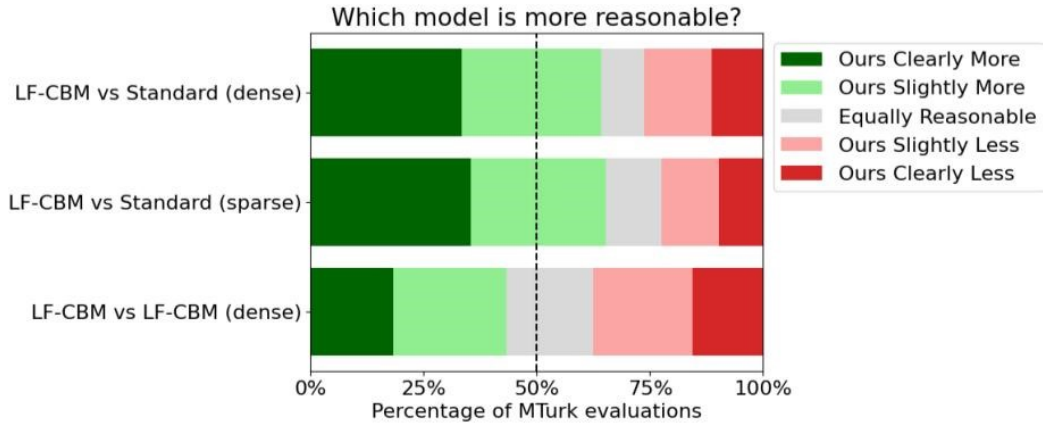


Figure E.15. Crowdsourced results for Task 2, without filtering any responses. We can see users strongly prefer LF-CBM explanations over standard models (the first two rows), but sparsity has little effect (the 3rd row).

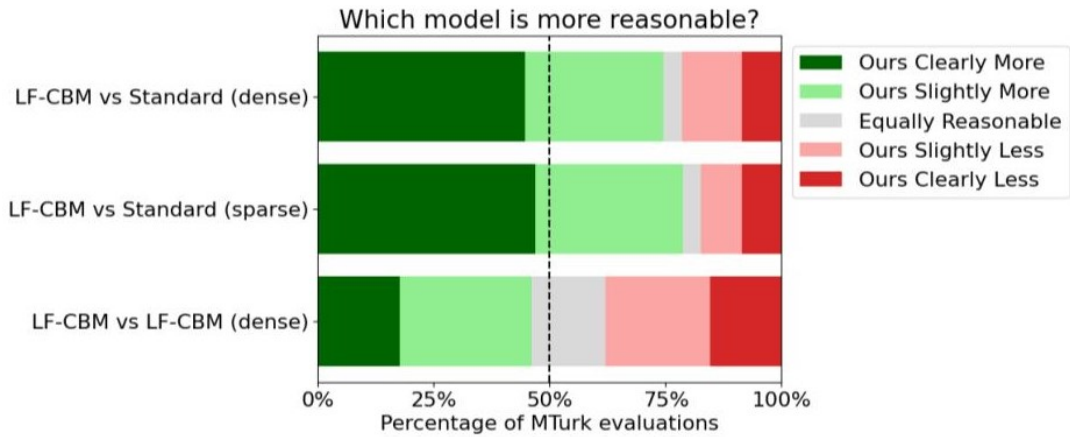


Figure E.16. Crowdsourced results for Task 2, where we filter out responses from any worker who gave at least one inconsistent response. We can see user preference LF-CBM explanations over standard models even more clearly (the first two rows).

Bibliography

- [1] Abubakar Abid, Mert Yuksekgonul, and James Zou. Meaningfully explaining model mistakes using conceptual counterfactuals. In *ICML*, 2022.
- [2] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *NeurIPS*, 31, 2018.
- [3] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. In *ICLR Workshop*, 2017.
- [4] Nicholas Bai, Rahul A Iyer, Tuomas Oikarinen, Akshay Kulkarni, and Tsui-Wei Weng. Interpreting neurons in deep vision networks with language models. *TMLR*, 2025.
- [5] David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. Rewriting a deep generative model. In *ECCV*, pages 351–369. Springer, 2020.
- [6] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *CVPR*, 2017.
- [7] 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. *PNAS*, 2020.
- [8] Umang Bhatt, Adrian Weller, and José M. F. Moura. Evaluating and aggregating feature-based model explanations. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3016–3022. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Main track.
- [9] Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models. <https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html>, 2023.
- [10] Tolga Bolukbasi, Adam Pearce, Ann Yuan, Andy Coenen, Emily Reif, Fernanda

Viégas, and Martin Wattenberg. An interpretability illusion for bert. *arXiv preprint arXiv:2104.07143*, 2021.

- [11] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models, 2021.
- [12] Judy Borowski, Roland Simon Zimmermann, Judith Schepers, Robert Geirhos, Thomas S. A. Wallis, Matthias Bethge, and Wieland Brendel. Exemplary natural images explain cnn activations better than state-of-the-art feature visualization. In *ICLR*, 2021.
- [13] Jordan Boyd-Graber, Samuel Carton, Shi Feng, Q Vera Liao, Tania Lombrozo, Alison Smith-Renner, and Chenhao Tan. Human-centered evaluation of explanations. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Tutorial Abstracts*, pages 26–32, 2022.
- [14] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermy, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. <https://transformer-circuits.pub/2023/monosemantic-features/index.html>.

- [15] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *NeurIPS*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.
- [16] Kirill Bykov, Laura Kopf, Shinichi Nakajima, Marius Kloft, and Marina MC Höhne. Labeling neural representations with inverse recognition. In *NeurIPS*, 2023.
- [17] Zhi Chen, Yijie Bei, and Cynthia Rudin. Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2(12):772–782, 2020.
- [18] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *ICLR*, 2024.
- [19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.
- [20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.
- [21] Nelson Elhage, Robert Lasenby, and Christopher Olah. Privileged bases in the transformer residual stream, 2023.
- [22] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. <https://transformer-circuits.pub/2021/framework/index.html>.
- [23] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.
- [24] Thomas Fel, Victor Boutin, Louis Béthune, Rémi Cadène, Mazda Moayeri, Léo Andéol, Mathieu Chalvidal, and Thomas Serre. A holistic approach to unifying automatic concept extraction and concept importance estimation. *NeurIPS*, 36, 2024.

- [25] Thomas Fel, Rémi Cadène, Mathieu Chalvidal, Matthieu Cord, David Vigouroux, and Thomas Serre. Look at the variance! efficient black-box explanations with sobol-based sensitivity analysis. *NeurIPS*, 34:26005–26014, 2021.
- [26] Thomas Fel, Agustin Picard, Louis Bethune, Thibaut Boissin, David Vigouroux, Julien Colin, Rémi Cadène, and Thomas Serre. Craft: Concept recursive activation factorization for explainability. In *CVPR*, pages 2711–2721, 2023.
- [27] Ruth Fong and Andrea Vedaldi. Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In *ICCV*, 2018.
- [28] Yossi Gandelsman, Alexei A Efros, and Jacob Steinhardt. Interpreting the second-order effects of neurons in clip. *ICLR*, 2025.
- [29] Robert Geirhos, Roland S. Zimmermann, Blair Bilodeau, Wieland Brendel, and Been Kim. Don’t trust your eyes: on the (un)reliability of feature visualizations. In *ICML*, 2024.
- [30] Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. *NeurIPS*, 32, 2019.
- [31] Gabriel† Goh, Nick† Cammarata, Chelsea Voss, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 2021. <https://distill.pub/2021/multimodal-neurons>.
- [32] Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus. <http://Skylion007.github.io/OpenWebTextCorpus>, 2019.
- [33] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- [34] Luke Guerdan, Solon Barocas, Ken Holstein, Hanna Wallach, Steven Wu, and Alexandra Chouldechova. Validating LLM-as-a-judge systems under rating indeterminacy. In *NeurIPS*, 2025.
- [35] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *ICML*, 2017.
- [36] Yoav Gur-Arieh, Roy Mayan, Chen Agassy, Atticus Geiger, and Mor Geva. Enhancing automated interpretability with output-centric feature descriptions. *ACL*, 2025.
- [37] Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. In

TMLR, 2023.

- [38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [39] Evan Hernandez, Sarah Schwettmann, David Bau, Teona Bagashvili, Antonio Torralba, and Jacob Andreas. Natural language descriptions of deep visual features. In *ICLR*, 2022.
- [40] Wenyi Hong, Weihang Wang, Ming Ding, Wenmeng Yu, Qingsong Lv, Yan Wang, Yean Cheng, Shiyu Huang, Junhui Ji, Zhao Xue, et al. Cogvlm2: Visual language models for image and video understanding. *arXiv preprint arXiv:2408.16500*, 2024.
- [41] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *NeurIPS*, volume 32. Curran Associates, Inc., 2019.
- [42] Jing Huang, Atticus Geiger, Karel D'Oosterlinck, Zhengxuan Wu, and Christopher Potts. Rigorously assessing natural language explanations of neurons. In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 317–331, 2023.
- [43] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, July 2021.
- [44] Aya Ismail, Tuomas Oikarinen, Amy Wang, Julius Adebayo, Samuel Stanton, Hector Corrada Bravo, Kyunghyun Cho, and Nathan Frey. Concept bottleneck language models for protein design. In *ICLR*, volume 2025, pages 56137–56169, 2025.
- [45] Amirhosein Javadi, Tuomas Oikarinen, Tara Javidi, and Tsui-Wei Weng. Ci-cbm: Class-incremental concept bottleneck model for interpretable continual learning. *TMLR*, 2026.
- [46] Neha Kalibhat, Shweta Bhardwaj, C Bayan Bruss, Hamed Firooz, Maziar Sanjabi, and Soheil Feizi. Identifying interpretable subspaces in image representations. In *ICML*, 2023.
- [47] Rémi Kazmierczak, Steve Azzolin, Eloïse Berthier, Anna Hedström, Patricia Delhomme, Nicolas Bousquet, Goran Frehse, Massimiliano Mancini, Baptiste Caramiaux, Andrea Passerini, et al. Benchmarking xai explanations with human-aligned evaluations. *arXiv preprint arXiv:2411.02470*, 2024.
- [48] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *ICML*, pages 2668–2677. PMLR, 2018.

- [49] Jenia Kim, Henry Maathuis, and Danielle Sent. Human-centered evaluation of explainable ai applications: a systematic review. *Frontiers in Artificial Intelligence*, 7:1456486, 2024.
- [50] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *ICML*, pages 5338–5348. PMLR, 2020.
- [51] Laura Kopf, Philine Lou Bommer, Anna Hedström, Sebastian Lapuschkin, Marina M-C Höhne, and Kirill Bykov. Cosy: Evaluating textual explanations of neurons. In *NeurIPS*, 2024.
- [52] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [53] Akshay Kulkarni, Ge Yan, Chung-En Sun, Tuomas Oikarinen, and Tsui-Wei Weng. Interpretable generative models through post-hoc concept bottlenecks. In *CVPR*, pages 8162–8171, 2025.
- [54] Biagio La Rosa, Leilani Gilpin, and Roberto Capobianco. Towards a fuller understanding of neurons with clustered compositional explanations. *NeurIPS*, 36, 2023.
- [55] Justin Lee, Tuomas Oikarinen, Arjun Chatha, Keng-Chi Chang, Yilan Chen, and Tsui-Wei Weng. The importance of prompt tuning for automated neuron explanations. In *NeurIPS ATTRIB Workshop*, 2023.
- [56] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. LLaVA-onevision: Easy visual task transfer. *TMLR*, 2025.
- [57] Max Losch, Mario Fritz, and Bernt Schiele. Interpretability beyond classification output: Semantic bottleneck networks, 2019.
- [58] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *NeurIPS*, 30, 2017.
- [59] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM ’10, page 1485–1488, New York, NY, USA, 2010. Association for Computing Machinery.
- [60] Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *NeurIPS*, 2022.
- [61] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*,

38(11):39–41, 1995.

- [62] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. In *ICLR*, 2022.
- [63] Arshia Soltani Moakhar, Eugenia Iofinova, Elias Frantar, and Dan Alistarh. Spade: Sparsity-guided debugging for deep neural networks. In *ICML*, 2024.
- [64] Jesse Mu and Jacob Andreas. Compositional explanations of neurons. *NeurIPS*, 33:17153–17163, 2020.
- [65] Jesse Mu and Jacob Andreas. Compositional explanations of neurons. In *NeurIPS*, 2020.
- [66] Geraldin Nanfack, Alexander Fulleringer, Jonathan Marty, Michael Eickenberg, and Eugene Belilovsky. Adversarial attacks on the interpretation of neuron activation maximization. In *AAAI*, 2024.
- [67] Tuomas Oikarinen, Subhro Das, Lam M Nguyen, and Tsui-Wei Weng. Label-free concept bottleneck models. In *ICLR*, 2023.
- [68] Tuomas Oikarinen and Tsui-Wei Weng. Clip-dissect: Automatic description of neuron representations in deep vision networks. In *ICLR*, 2023.
- [69] Tuomas Oikarinen and Tsui-Wei Weng. Linear explanations for individual neurons. In *ICML*, 2024.
- [70] Tuomas Oikarinen, Ge Yan, Akshay Kulkarni, and Tsui-Wei Weng. Beyond top activations: Efficient and reliable crowdsourced evaluation of automated interpretability. In *CVPR*, 2026.
- [71] Tuomas Oikarinen, Ge Yan, and Tsui-Wei Weng. Evaluating neuron explanations: A unified framework with sanity checks. In *ICML*, 2025.
- [72] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. <https://distill.pub/2020/circuits/zoom-in>.
- [73] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.
- [74] OpenAI. GPT-4o system card. <https://openai.com/index/gpt-4o-system-card/>, 2024.
- [75] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. In *BMVC*, page 151. BMVA Press, 2018.

- [76] Hieu Pham, Zihang Dai, Golnaz Ghiasi, Hanxiao Liu, Adams Wei Yu, Minh-Thang Luong, Mingxing Tan, and Quoc V Le. Combined scaling for zero-shot transfer learning. *arXiv preprint arXiv:2111.10050*, 2021.
- [77] 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. In *ICML*, 2021.
- [78] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.
- [79] Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *Journal of Machine Learning Research*, 11(43):1297–1322, 2010.
- [80] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ”why should i trust you?” explaining the predictions of any classifier. In *KDD*, 2016.
- [81] Christian Robert and George Casella. *Monte Carlo Statistical Methods, second edition*. Springer New York, 2004.
- [82] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistic Surveys*, 16, 2022.
- [83] Hassan Sajjad, Nadir Durrani, and Fahim Dalvi. Neuron-level interpretation of deep nlp models: A survey. In *TACL*, 2022.
- [84] Wojciech Samek, Grégoire Montavon, Sebastian Lapuschkin, Christopher J Anders, and Klaus-Robert Müller. Explaining deep neural networks and beyond: A review of methods and applications. *Proceedings of the IEEE*, 109(3):247–278, 2021.
- [85] Sarah Schwettmann, Tamar Shaham, Joanna Materzynska, Neil Chowdhury, Shuang Li, Jacob Andreas, David Bau, and Antonio Torralba. Find: A function description benchmark for evaluating interpretability methods. *NeurIPS*, 36:75688–75715, 2023.
- [86] Tamar Rott Shaham, Sarah Schwettmann, Franklin Wang, Achyuta Rajaram, Evan Hernandez, Jacob Andreas, and Antonio Torralba. A multimodal automated interpretability agent. In *ICML*, 2024.
- [87] Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo

- Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. Dinov3, 2025.
- [88] Chandan Singh, Aliyah R. Hsu, Richard Antonello, Shailee Jain, Alexander G. Huth, Bin Yu, and Jianfeng Gao. Explaining black box text modules in natural language with language models, 2023.
 - [89] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *NeurIPS*. Curran Associates, Inc., 2016.
 - [90] Robyn Speer and Catherine Havasi. Conceptnet 5: A large semantic network for relational knowledge. *The people’s web meets NLP, theory and applications of natural language processing*, pages 161–176, 02 2013.
 - [91] Anvita A Srinivas, Tuomas Oikarinen, Divyansh Srivastava, Wei-Hung Weng, and Tsui-Wei Weng. Sand: Enhancing open-set neuron descriptions through spatial awareness. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2993–3002. IEEE, 2025.
 - [92] Divyansh Srivastava, Tuomas Oikarinen, and Tsui-Wei Weng. Corrupting neuron explanations of deep visual features. In *ICCV*, 2023.
 - [93] Divyansh Srivastava, Ge Yan, and Tsui-Wei Weng. Vlg-cbm: Training concept bottleneck models with vision-language guidance. *NeurIPS*, 2024.
 - [94] Nishant Subramani, Nivedita Suresh, and Matthew E Peters. Extracting latent steering vectors from pretrained language models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 566–581, 2022.
 - [95] Chung-En Sun, Tuomas Oikarinen, Berk Ustun, and Tsui-Wei Weng. Concept bottleneck large language models. In *ICLR*, volume 2025, pages 89371–89411, 2025.
 - [96] Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024.
 - [97] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understand-

ing, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

- [98] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.
- [99] Alvin Wan, Lisa Dunlap, Daniel Ho, Jihan Yin, Scott Lee, Suzanne Petryk, Sarah Adel Bargal, and Joseph E. Gonzalez. {NBDT}: Neural-backed decision tree. In *ICLR*, 2021.
- [100] Sheng-Yu Wang, David Bau, and Jun-Yan Zhu. Rewriting geometric rules of a gan. *ACM Transactions on Graphics (TOG)*, 41(4):1–16, 2022.
- [101] Zeyu Wang, Berthy Feng, Karthik Narasimhan, and Olga Russakovsky. Towards unique and informative captioning of images. In *ECCV*, pages 629–644. Springer, 2020.
- [102] Peter Welinder, Steve Branson, Pietro Perona, and Serge Belongie. The multidimensional wisdom of crowds. In *NeurIPS*, volume 23, 2010.
- [103] Eric Wong, Shibani Santurkar, and Aleksander Madry. Leveraging sparse linear layers for debuggable deep networks. In *ICML*, 2021.
- [104] Sin-Han Yang, Tuomas Oikarinen, and Tsui-Wei Weng. Concept-driven continual learning. *TMLR*, 2024.
- [105] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. In *TMLR*, 2022.
- [106] Qihang Yu, Ju He, Xueqing Deng, Xiaohui Shen, and Liang-Chieh Chen. Convolutions die hard: Open-vocabulary segmentation with single frozen convolutional clip. In *NeurIPS*, 2023.
- [107] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021.
- [108] Mert Yuksekgonul, Maggie Wang, and James Zou. Post-hoc concept bottleneck models. *ICLR*, 2023.
- [109] Mateo Espinosa Zarlenga, Pietro Barbiero, Gabriele Ciravegna, Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, Zohreh Shams, Frederic Precioso, Stefano Melacci, Adrian Weller, Pietro Lio, and Mateja Jamnik. Concept embedding models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *NeurIPS*, 2022.

- [110] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833. Springer, 2014.
- [111] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *ICCV*, 2023.
- [112] Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer. Lit: Zero-shot transfer with locked-image text tuning. In *CVPR*, pages 18123–18133, 2022.
- [113] Ruihan Zhang, Prashan Madumal, Tim Miller, Krista A Ehinger, and Benjamin IP Rubinstein. Invertible concept-based explanations for cnn models with non-negative concept activation vectors. In *AAAI*, volume 35, pages 11682–11690, 2021.
- [114] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. In *ICLR*, 2015.
- [115] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.
- [116] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Interpretable basis decomposition for visual explanation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 119–134, 2018.
- [117] Roland S. Zimmermann, Thomas Klein, and Wieland Brendel. Scale alone does not improve mechanistic interpretability in vision models. In *NeurIPS*, 2023.