SUNY: A Visual Interpretation Framework for Convolutional Neural Networks from a Necessary and Sufficient Perspective

Xiwei Xuan[1,*] Ziquan Deng[1] Hsuan-Tien Lin[2] Zhaodan Kong[1] Kwan-Liu Ma[1]
[1]University of California, Davis [2]National Taiwan University [*]Corresponding Author
{xwxuan, ziqdeng}@ucdavis.edu htlin@csie.ntu.edu.tw {zdkong, klma}@ucdavis.edu

A. Appendix

This appendix section is organized as follows:

- Sec. A.1 presents the flexibility of *SUNY* to consider either a feature map or a model filter in a convolutional layer as a cause for analysis.
- Sec. A.2 provides a comparison between *SUNY* and previous SHAP image analyses [3].
- Sec. A.3 provides a more detailed description of *SUNY* implementation.
- Sec. A.4 provides more *SUNY* examples for qualitative evaluations.
- Sec. A.5 presents the training setting for our experiments.

A.1. A General Framework

As mentioned at the end of Section 2.1, our proposed framework provides the flexibility to consider either a feature map or a model filter in a convolutional layer as a cause for analysis. We differentiate between these two options as *SUNY*-feature and *SUNY*-filter in the Appendix, and provide an overview of the *SUNY*-filter framework in Fig. A1.

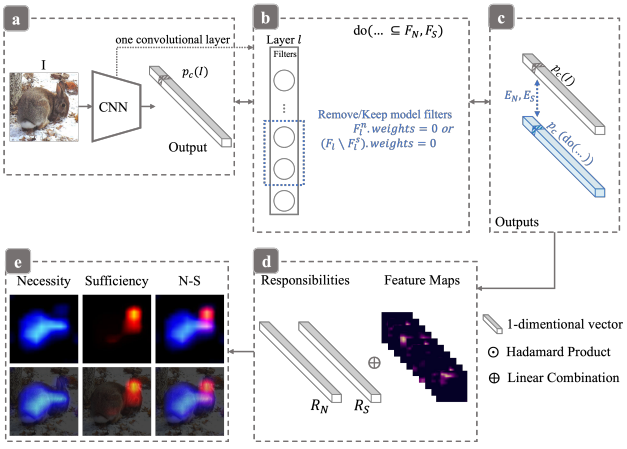

Figure A1. Overview of *SUNY*-filter framework.

A.2. Comparison with SHAP Methods

For our requirement $G1$, i.e. "to measure the importance of each individual cause in a group of coordinating causes," shapely value [2] provides us a rational optimal solution to quantify the marginal contributions across diverse coalitions:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [v(S \cup \{i\}) - v(S)], \tag{A1}$$

where ϕ_i is the Shapley value for an element i, F is the set of all elements, S is a subset of F that does not include element i, $v(S)$ is the value function that gives the total payoff that the subset of elements S can obtain by themselves. In our work, $E_N(\cdot)$ and $E_S(\cdot)$ (Eqn.s (1) and (2), respectively) can be seen as two value functions with different causal semantics. In practical applications, however, Equation A1 encounters a widely recognized issue of computational complexity. The primary reason for this issue is attributable to the excessive size of the total set S. Considering the unique attributes of the CNN model as applied to image processing, we have implemented the following modifications:

- **Constraining the scope of the subset S.** Rather than blindly covering all subsets and elements, we tend to focus on more necessary (sufficient) ones. We retain individuals with a higher $E_N(i)$ (/$E_S(i)$) to form F_N (/F_S) to constrain the scopes of the subset S and the element i.
- **Combine the model to select element i effectively.** In previous SHAP image analysis [3], images are divided into uniformly sized patches, each treated as element i for Shapley value calculation. This method limits the use of smaller patches due to high computational demands, leading to SHAP saliency maps that lack fine-grained importance distinctions, as illustrated in the Fig. A2. Our method, when integrated with the model, allows for the selection of either feature maps or filters as the object of analysis, thereby facilitating the provision of fine-grained visualization results.

Furthermore, we provide value functions (Eqns.(1) and (2))

with distinct causal interpretations to aptly characterize the significance derived from N and S.

Figure A2. SHAP[3] explanation example.

A.3. Algorithm Details

We present the implementation of *SUNY* in Alg. 1. Following the definitions in Sec. 2.1 in the main paper, *SUNY* provides causality-driven CNN visual explanations regarding input features or model filters as hypothesized causes, respectively, represented by the cause type E in Alg. 1 with values *"feature"* or *"filter"*. The corresponding explanations are *SUNY*-feature and *SUNY*-filter. We first define $p_c(\cdot)$ as a function to calculate the model's prediction probability w.r.t. a class c for the input denoted by $\cdot$, as shown in line 1 of Alg. 1. Next, in line 3, we construct F_N and F_S, where F_N is the set of hypothetical single causes f_n with relatively higher **N Effect**, $E_N(f_n)$, (refer to Eqn. (1) in Sec. 2.1 of the main paper). Similarly, F_S contains all hypothetical single causes f_s with higher **S Effect**, $E_S(f_s)$, (refer to Eqn. (2) in the main paper). We then calculate **N-S Responsibilities** for every single cause in F_N and F_S following lines 5 - 20 (refer to Eqn. (3), (4) in the main paper). Specifically, for *SUNY*-feature (lines 6 - 11), we upsample and normalize the feature maps, A_l^n and A_l^s, and use the generated $mask$ as a feature extractor to intervene on input features from the image I. The intervention $\mathrm{do}(F \setminus F_*)$ (removing F_*) is realized by the Hadamard product, $(I \bigodot (1 - mask))$. Similarly, $\mathrm{do}(F_*)$ (keeping F_*) is implemented as $(I \bigodot mask)$. *SUNY*-filter (lines 13 - 20) removes and keeps the hypothesized causes by filter pruning, which means setting the corresponding filters' weights to zero. Line 14 and line 18 correspond to $\mathrm{do}(F \setminus F_l^n)$ and $\mathrm{do}(F_l^s)$, respectively. After calculating **N-S Responsibilities** for all single causes, in lines 21 and 22, we obtain small saliency maps by $Relu(\sum {R_N}^i A_l^i)$, $Relu(\sum {R_S}^i A_l^i)$ and then upsample and normalize them to get the final saliency maps. The operation $norm$ represents the min-max normalization w.r.t. each single map, $norm(X) = \frac{X - min(X)}{max(X) - min(X)}$, and $upsample$ represents the bilinear interpolation.

Algorithm 1 *SUNY*: Causal Explanation of CNN

Require: Image I, model M, layer l, class c, cause type E
Ensure: N-S saliency maps: N_{map}, S_{map}

1: $p_c(\cdot) = \mathrm{Softmax}(M(\cdot))[c]$ ▷ prediction probability on c
2: $A_l \leftarrow M_l(I)$ ▷ feature maps of the layer l
3: $F_N, F_S \leftarrow \mathrm{getHypCauses}(I, M, l, E)$
4: $R_N, R_S \leftarrow zeros(A_l.shape[0])$ ▷ initialize responsibilities
5: **if** E is "$feature$" **then**
6: **for** A_l^n **in** F_N **do**
7: $mask \leftarrow \mathrm{norm}(\mathrm{upsample}(A_l^n))$
8: Compute $R_N(A_l^n)$ based on Eqn.(3)
9: **for** A_l^s **in** F_S **do**
10: $mask \leftarrow \mathrm{norm}(\mathrm{upsample}(A_l^s))$
11: Compute $R_S(A_l^s)$ based on Eqn.(4)
12: **else if** E is "$filter$" **then**
13: **for** F_l^n **in** F_N **do**
14: $M^n \leftarrow \mathrm{pruneFilters}(M, F_l^n)$
15: $p_c^n(\cdot) = \mathrm{Softmax}(M^n(\cdot))[c]$
16: Compute $R_N(F_l^n)$ based on Eqn.(3)
17: **for** F_l^s **in** F_S **do**
18: $M^s \leftarrow \mathrm{pruneFilters}(M, (F_l \setminus F_l^s))$
19: $p_c^s(\cdot) = \mathrm{Softmax}(M^s(\cdot))[c]$
20: Compute $R_S(F_l^s)$ based on Eqn.(4)
21: $N_{map} = \mathrm{norm}(\mathrm{upsample}(\mathrm{Relu}(\sum {R_N}^i A_l^i)))$
22: $S_{map} = \mathrm{norm}(\mathrm{upsample}(\mathrm{Relu}(\sum {R_S}^i A_l^i)))$
23: **return** N_{map}, S_{map}

A.4. Additional Heatmap Examples

In this section, we provide more examples of semantic evaluations, where Fig. A3 and Fig. A4 present comparisons between *SUNY* and other methods, and Fig. A5 include more causal explanation examples to demonstrate the usefulness of *necessity* and *sufficiency* ((a)(b)(c)) and textual explanation examples (as discussed below) to further examine more semantically-meaningful explanations with *SUNY*.

Textual explanations with *SUNY*. Recent research indicates that filters in a convolutional layer act as concept detectors [6, 7] and describes each filter with text [1, 5]. To examine visual-textual explanations with *SUNY*, we adopt [1] to automatically name the filters in a convolutional layer and present the top filters based on **N-S Responsibilities** (R_N, R_S) provided by *SUNY*, where higher R_N, R_S mean the corresponding filters are more necessary, sufficient, respectively. Fig. A5(d) shows some examples of visual and textual explanations with *SUNY*-filter, where we sort filters in descending order of the normalized R_N and R_S respectively, and present the top filters' textual explanations. For each image, *SUNY* visualizations highlight regions that are *Sufficient* or *Necessary* for a specific prediction, and the complementary textual explanations for the top *N* and *S* filters further explain why the corresponding regions are essential for the class. By discussing the textual explanations with *SUNY*, we extend the usefulness of our technique in the ability to provide both intuitive and semantically-meaningful explanations.

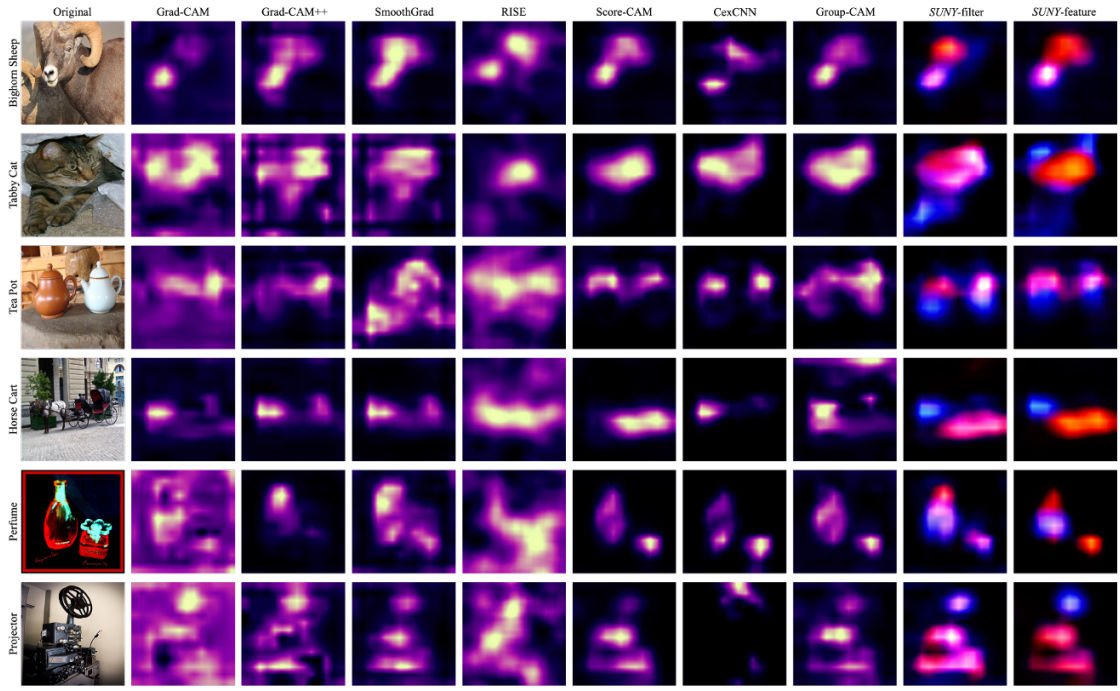

Figure A3. Visual comparison of saliency maps from different methods based on a VGG16 trained on ILSVRC. Each row is corresponding to one image from the ILSVRC validation set that is classified correctly by the model. The specified class for generating explanations is the predicted class, which is shown at the beginning of each row.

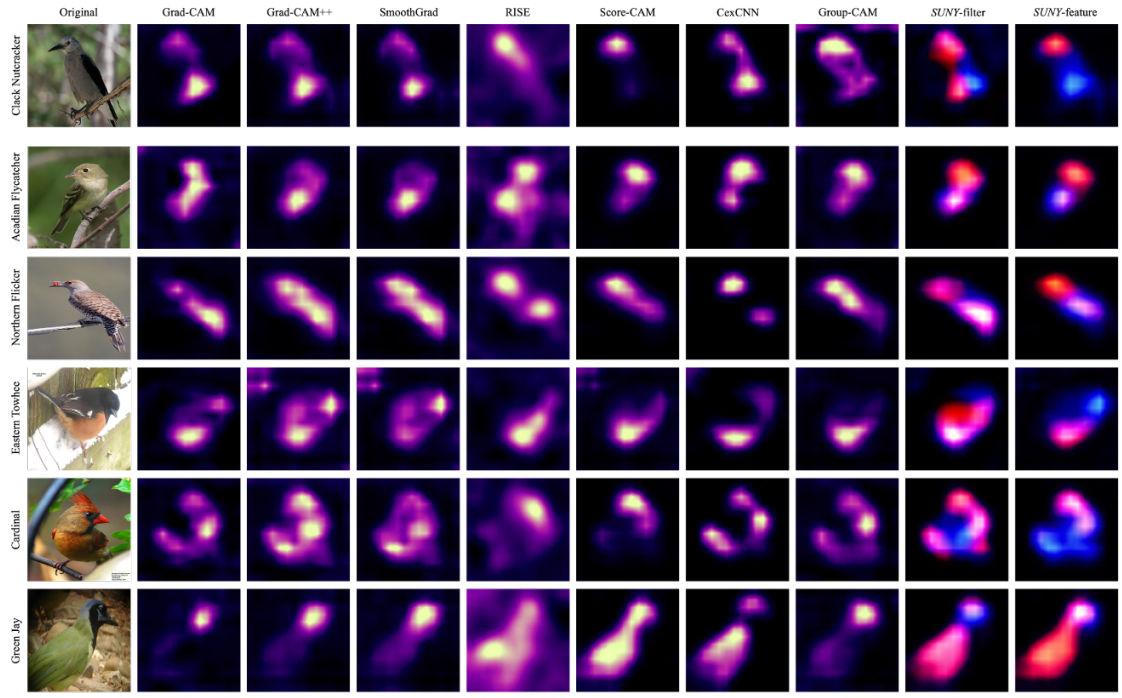

Figure A4. Visual comparison of saliency maps from different methods based on a VGG16 trained on CUB. Each row is corresponding to one image from the CUB validation set that is classified correctly by the model. The specified class for generating explanations is the predicted class, which is shown at the beginning of each row.

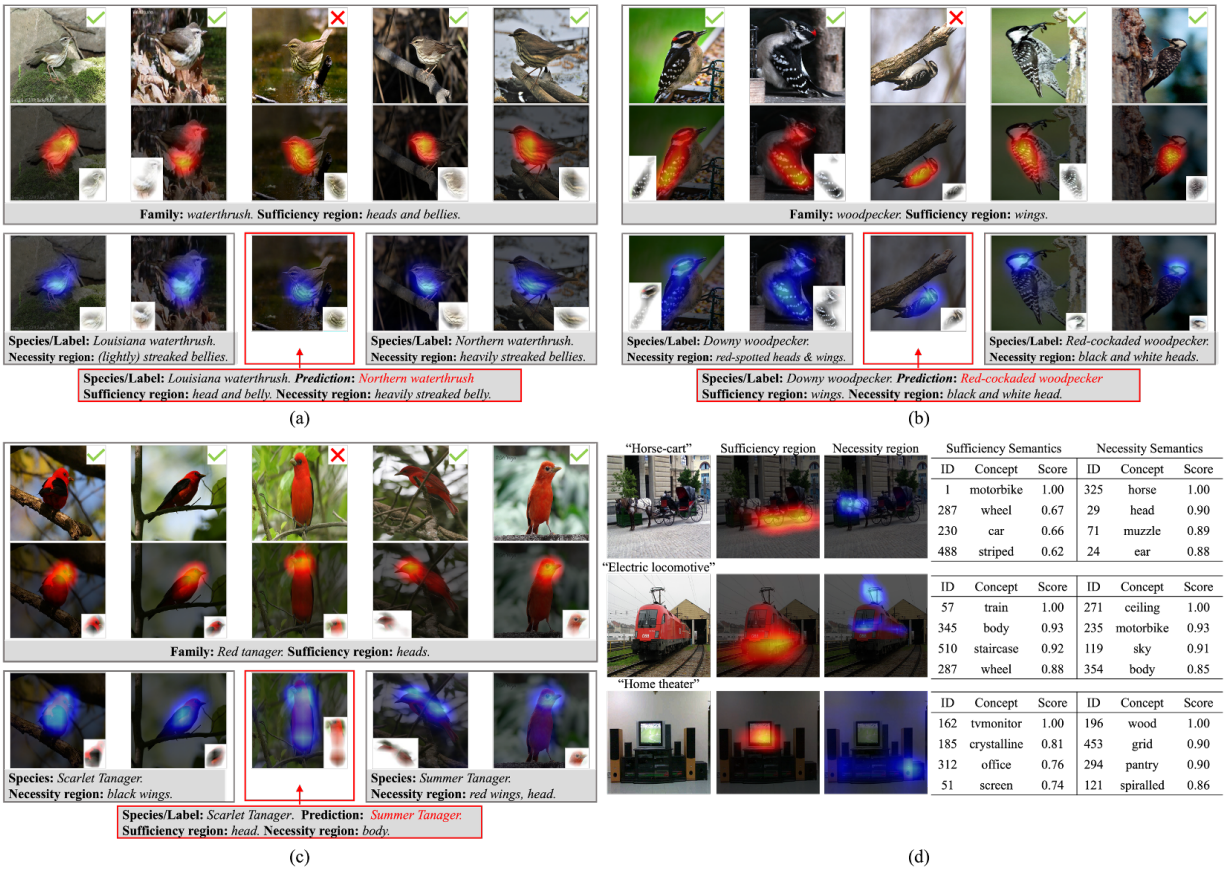

Figure A5. (a)-(c) Semantic evaluation of *SUNY* explanations for a VGG16 trained on CUB for bird species classification. The bird images in the first row are from four bird species belonging to two families and the correct/incorrect predictions are marked by ✔ and ✖, respectively. For the two images marked by ✖, the model mistakes the actual species with the other species under the same family. Each column corresponds to one image; the second and third rows are *sufficiency* and *necessity* heatmaps, respectively. The small image in the bottom corner of each heatmap presents the highlighted image portion. (d) Examples of visual and textual explanations with *SUNY*-filter for a VGG16 trained on ILSVRC. For textual explanations, we provide top filters corresponding to the highest R_S and R_N, respectively.

A.5. Experiment Details

We implement *SUNY* and the seven aforementioned visual explanation methods in Python using PyTorch framework [4]. Specifically, we run the official or publicly available code of other methods as the results at the same data scale are unavailable. The platform is equipped with two NVIDIA RTX 3090 GPUs. Unless explicitly stated, the comparisons discussed in this section are conducted following the same settings, including –

(1) Images are resized and converted to the RGB format, transformed to tensors, and normalized to the range of $[0, 1]$.

(2) For every single input image across three datasets, visual explanation results are generated to explain every model with respect to (w.r.t.) every ground-truth class.

(3) For images with multiple ground-truth classes, the overall performance is measured by the mean quantification score across all classes.

Bird Species Classifier Training Details.

We trained bird species classifiers using CUB-200-2011 (CUB) with pre-trained VGG16, ResNet50, and Inception_v3, respectively. All models are retrieved from the TorchVision model zoo. We use the cross entropy loss and the SGD optimizer with a momentum of 0.9. The learning rate is 0.001 for VGG16 and ResNet50, and 0.0001 for Inception_v3.

Saliency Attack Details.

In the saliency attack evaluation, the image regions highlighted by a specific visual explanation method are corrupted using random Gaussian noise with mean $\mu = 0$, variance $\sigma^2 = 0.03$. Specifically, we generate a noise matrix $noise$ of the same size as the original input image I, where $noise[i, j] \sim \mathcal{N}(0,\ 0.03)$. For each visual explanation method, we have its saliency map as a matrix, where $map[i, j] \in [0, 1]$. The final image that we feed into the

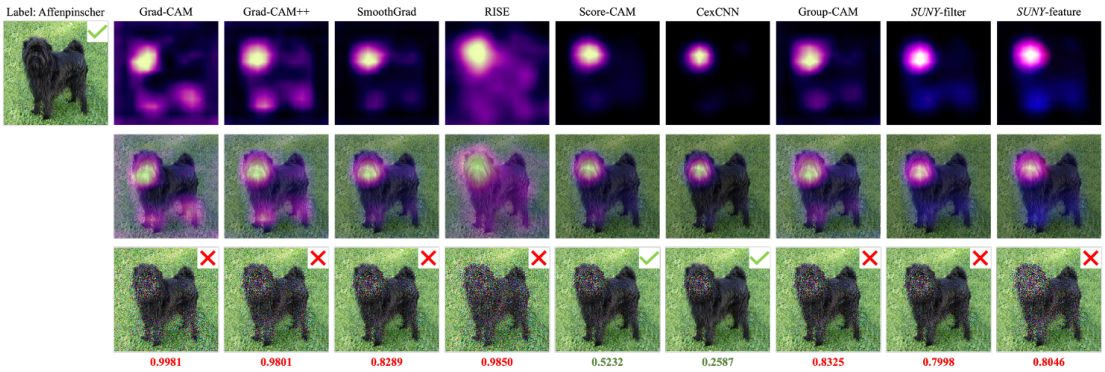

Figure A6. Saliency attack results based on different visual explanations. The original image on the top left is from the validation set of ILSVRC with the label `Affenpinscher`, which is classified correctly by the model. For the image grid, from the first to third row: heatmap, heatmap on the image, image with noise on the heatmap-highlighted region. The ✔ and ✖ indicate the model's correct/incorrect prediction of the corresponding image. Each column corresponds to one visual explanation. The number below each column is the size of the region highlighted by the saliency map ($Size_{map}$), where red indicates a successful saliency attack and green indicates a failed attack (For successful attacks, lower $Size_{map}$ is better).

model is generated by $I' = I + noise \bigodot map$. Recalling our paper, the size of the region highlighted by the saliency map is calculated by $Size_{map} = AttackSize = \frac{\sum_{i,j}(1_{true}map[i,j]\neq 0)}{map.h \times map.w}$.

In Fig. A6, we provide examples corresponding to one original image, where we add noise to the saliency regions and test whether such noise can fool the model. The original image shown in Fig. A6 is classified correctly. From the ✔ and ✖ marks over the images with noise, we can find that the saliency attack is successful for most explanations (wrong classifications) except Score-CAM and CexCNN (correct classifications). Moreover, for the explanations corresponding to successful saliency attacks, *SUNY*-filter and *SUNY*-feature have the smallest $Size_{map}$, indicating the most "minimal and essential" attacks.

The entire *SUNY* codebase and experiment setup will be made publicly available upon the publication of this paper.

References

[1] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017. 2

[2] Sergiu Hart. Shapley value. In *Game theory*, pages 210–216. Springer, 1989. 1

[3] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017. 1, 2

[4] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 4

[5] Yu Yang, Seungbae Kim, and Jungseock Joo. Explaining deep convolutional neural networks via latent visual-semantic filter attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8333–8343, 2022. 2

[6] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 2

[7] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014. 2