Supplementary Materials for "Understanding Deep Networks via Extremal Perturbations and Smooth Masks"

Ruth Fong[†] University of Oxford[*]
Mandela Patrick[†] University of Oxford
Andrea Vedaldi Facebook AI Reseach

Contents

List of Figures

[*]Work done as a contractor in FAIR. [†] equal contributions.

1. Proof for Lemma 1

Proof. For the first property $\boldsymbol{m}(u) = \max_{v\in\Omega} \boldsymbol{k}(u-v)\bar{\boldsymbol{m}}(v) \geq \boldsymbol{k}(u-u)\bar{\boldsymbol{m}}(u) = \bar{\boldsymbol{m}}(u)$. For the second property, if $\boldsymbol{k}$ is Lipschitz continuous with constant K, then $|\boldsymbol{k}(u) - \boldsymbol{k}(u')| \leq K\|u-u'\|$. We whish to bound $|\boldsymbol{m}(u) - \boldsymbol{m}(v)|$. Assuming without loss of generality that $\boldsymbol{m}(u) \geq \boldsymbol{m}(v)$, we have

$$
\begin{aligned}
|\boldsymbol{m}(u) - \boldsymbol{m}(v)| = \boldsymbol{m}(u) - \boldsymbol{m}(v) = \\
\max_{v\in\Omega} \boldsymbol{k}(u-v)\bar{\boldsymbol{m}}(v) - \max_{v'\in\Omega} \boldsymbol{k}(u'-v')\bar{\boldsymbol{m}}(v') \leq \\
\max_{v\in\Omega} (\boldsymbol{k}(u-v)\bar{\boldsymbol{m}}(v) - \boldsymbol{k}(u-v')\bar{\boldsymbol{m}}(v)) \leq \\
\max_{v\in\Omega} \boldsymbol{m}(v) \cdot |\boldsymbol{k}(u-v) - \boldsymbol{k}(u-v')| \leq \\
\max_{v\in\Omega} \boldsymbol{m}(v) \cdot K\|u'-u\| \leq K\|u'-u\|.
\end{aligned}
$$

□

2. Visualizations and Details for Area Constraint and Parameteric Family of Smooth Masks

We have included a few illustrations of our technical innovations. Figure 1 visualizes our novel area constraint loss. Figure 2 demonstrates how we use the max-convolution in practice to generate smooth masks.

2.1. Differentiability of vecsort

The vecsort function is differentiable for our purposes. We use the PyTorch sort function, which returns a tensor of sorted values and a tensor of sorted indices; the sorted values are differentiable but the indices are not. The backward function for the sorted values tensor simply "unsorts"

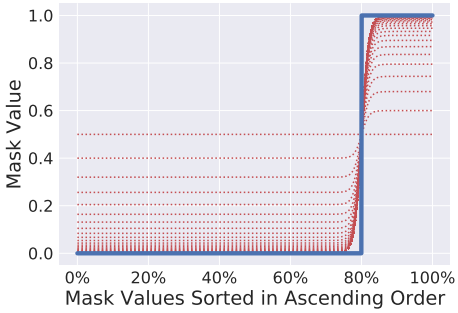

Figure 1: **Area constraint.** The blue line is the reference vector $\mathbf{r}_a$ representing the target area a as a proportion of "on" versus "off" pixels (here $a = 20\%$). The red dotted lines represent mask values in ascending, sorted order, $\text{vecsort}(\boldsymbol{m})$, at different points of the optimization, staring by initializing $\boldsymbol{m}$ to $1/2$. Over time, the optimization encourages the mask to match the target area. Even in the limit, there is a slight difference between $\mathbf{r}_a$ as $\boldsymbol{m}$ as the latter is forced to be smooth at the boundaries; this has the further beneficial effect of minimizing the boundary length, encouraging more compact masks.

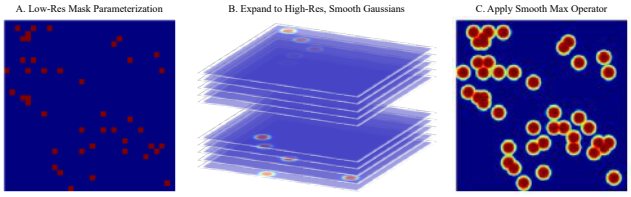

Figure 2: **Generating smooth masks.** a. We first define a low resolution parameterization mask, $\bar{\boldsymbol{m}} \in [0,1]^{H \times W}$ (shown as a binary mask in this illustration). b. We then represent each location in $\bar{\boldsymbol{m}}$ as a high resolution, smooth Gaussian, weighted by the value in $\bar{\boldsymbol{m}}$. c. Finally, we apply the smooth max operator across all locations to get our final mask.

the gradient being backpropagated. That is, for function $\text{sorted}, \text{indices} = \text{torch.sort}(x)$, the backward function is as follows below; we only use the sorted values in our area constraint (Figure 1a).

```
def sort_backward(ctx, grad_output):
  # Get indices for sorted to unsorted.
  _, indices_ = torch.sort(ctx.indices)
  return grad_output[indices_], None
```

3. Comparison Visualizations

3.1. Comparison with other visualization methods

Figure 3 and fig. 4 show comparisons between our method and other popular visualization methods, such as gradient [5], guided backprop [6], Grad-CAM [4], and RISE [3].

3.2. Comparison with [2]

Figure 5 show more examples comparing our method with [2]. We compare our masks generated using VGG16 trained in PyTorch against their masks generated using GoogLeNet trained in Caffe, as that is what their method was optimized for. One can see in fig. 5 that [2]'s masks consistently are one connected component and are unable to identify multiple, distinct objects of the same class. One can also note the instability of [2] in some of their learned mask, most likely in part due to the tradeoff between their regularization terms (i.e., L1 sparsity term and total variation smoothness term).

3.3. Area Growth

The following figures include area growth progressions for the first 50 validation images in ImageNet: fig. 6, fig. 7, fig. 8, fig. 9. The bar graph visualizes $\Phi(\boldsymbol{m}_a \odot \boldsymbol{x})$ as a normalized fraction of $\Phi_0 = \Phi(\boldsymbol{x})$ (i.e., class score on original image) and $\Phi_{\text{min}} = \Phi(\boldsymbol{m}_0 \odot \boldsymbol{x})$ (i.e., class score on fully perturbed image) and saturates after exceeding Φ_0 by 25% as follows:

$$v = \max\left(\frac{\Phi(\boldsymbol{m}_a \odot \boldsymbol{x}) - \Phi_{\text{min}}}{\Phi_0 - \Phi_{\text{min}}}, 1.25\right) \quad (1)$$

3.4. Sanity Checks [1]

Adebayo et al. [1] proposed several "sanity checks" for evaluating the specificity of attribution methods to model weights. We show qualitative experiments for our method in fig. 10 and fig. 11 and for other methods in fig. 12 and fig. 13 for [1]'s model parameter randomization test. Cascading randomization successively randomizes a models' learnable weights, from the end of a model (e.g., fc8 in VGG16) to the beginning of a model (e.g., conv1_1 in VGG16), whereas independent randomization randomizes a single layer's random weights. When randomizing a layer's weights, we draw from same distribution that we used to initialize the model during training.

Implementation Details. We use the "hybrid" formulation instead of "preservation" formulation used elsewhere in the paper. The input images were resized to (224, 224).

3.5. Pointing Game Examples

Figure 14 and fig. 15 show a few examples of our masks being discriminative for different classes on PASCAL and COCO respectively.

4. Visualizations of Channel Attribution at Intermediate Layers

Figure 16, fig. 17, and fig. 18 show saliency heatmaps and feature inversions of our channel attribution masks. By comparing the difference in feature inversions between the original and perturbed activations, we can identify the salient features that our method highlights. To our knowledge, this is the first work that shows side-by-side "diffs" of feature inversions.

References

[1] Julius Adebayo, Justin Gilmer, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Proc. NeurIPS*, 2018. 1, 2

[2] Ruth Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proc. ICCV*, 2017. 1, 2, 6

[3] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. In *Proc. BMVC*, 2018. 2

[4] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proc. ICCV*, 2017. 2

[5] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proc. ICLR workshop*, 2014. 2

[6] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. 2015. 2

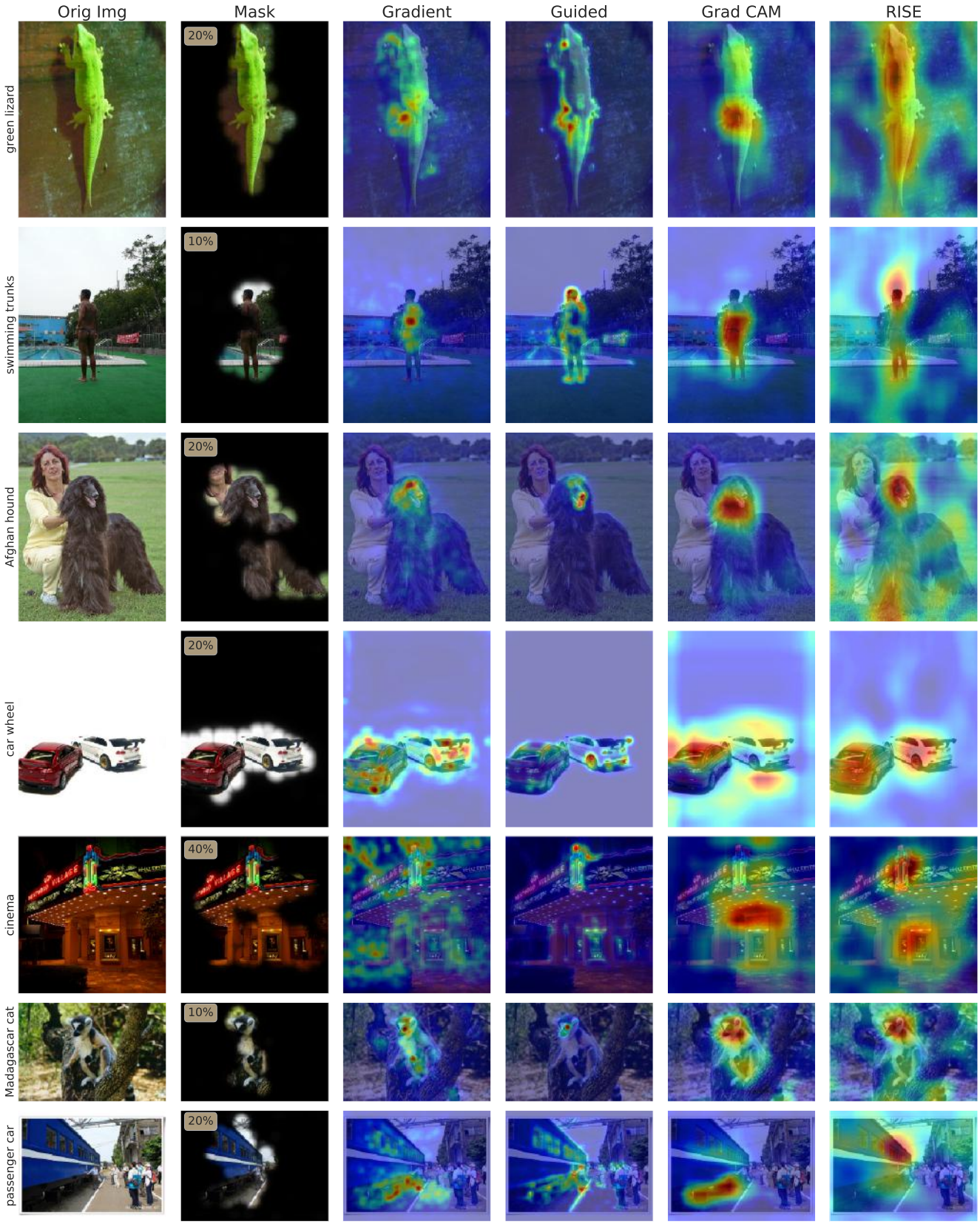

Figure 3: **Comparison with other visualization methods.**

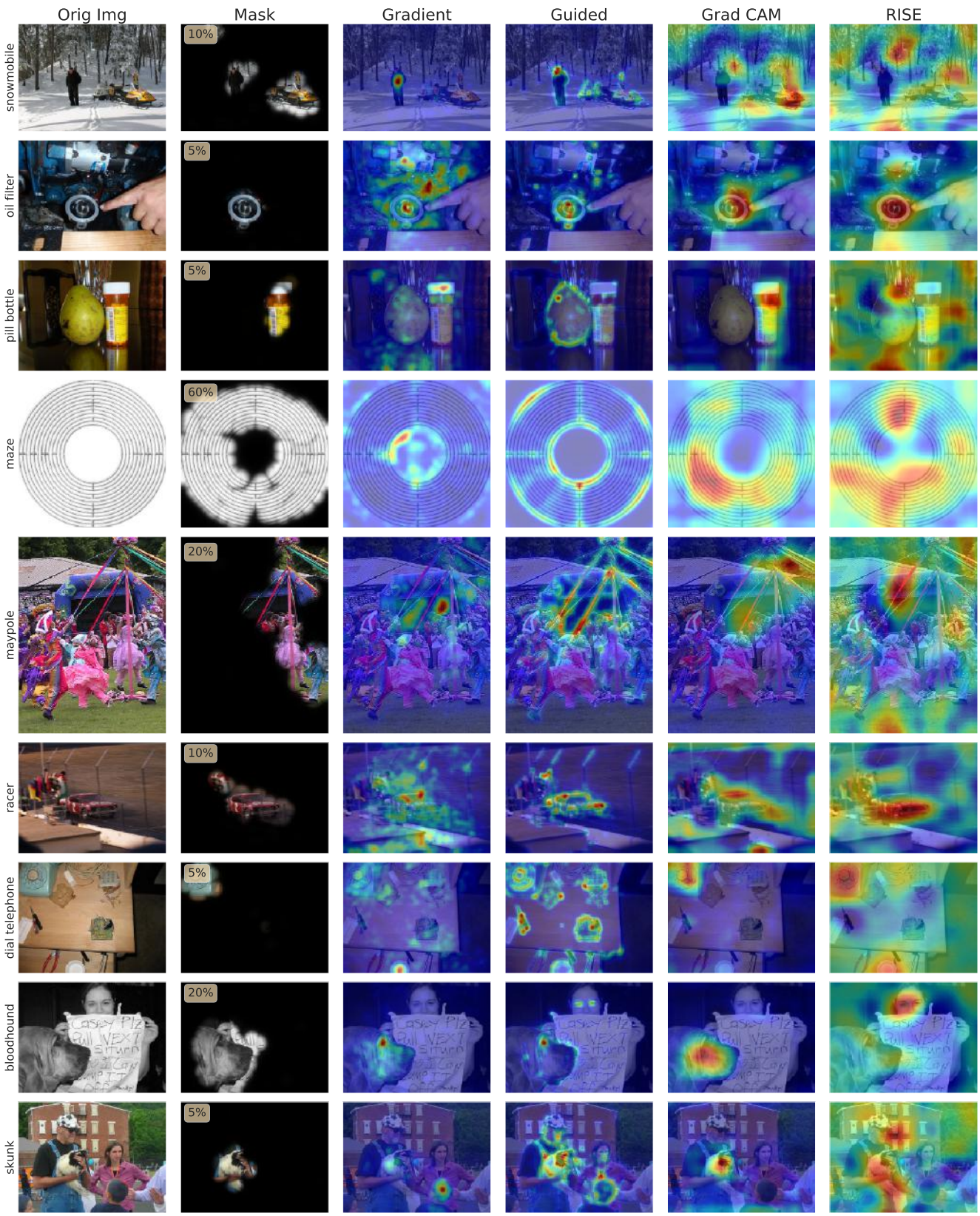

Figure 4: **Comparison with other visualization methods (more examples).**

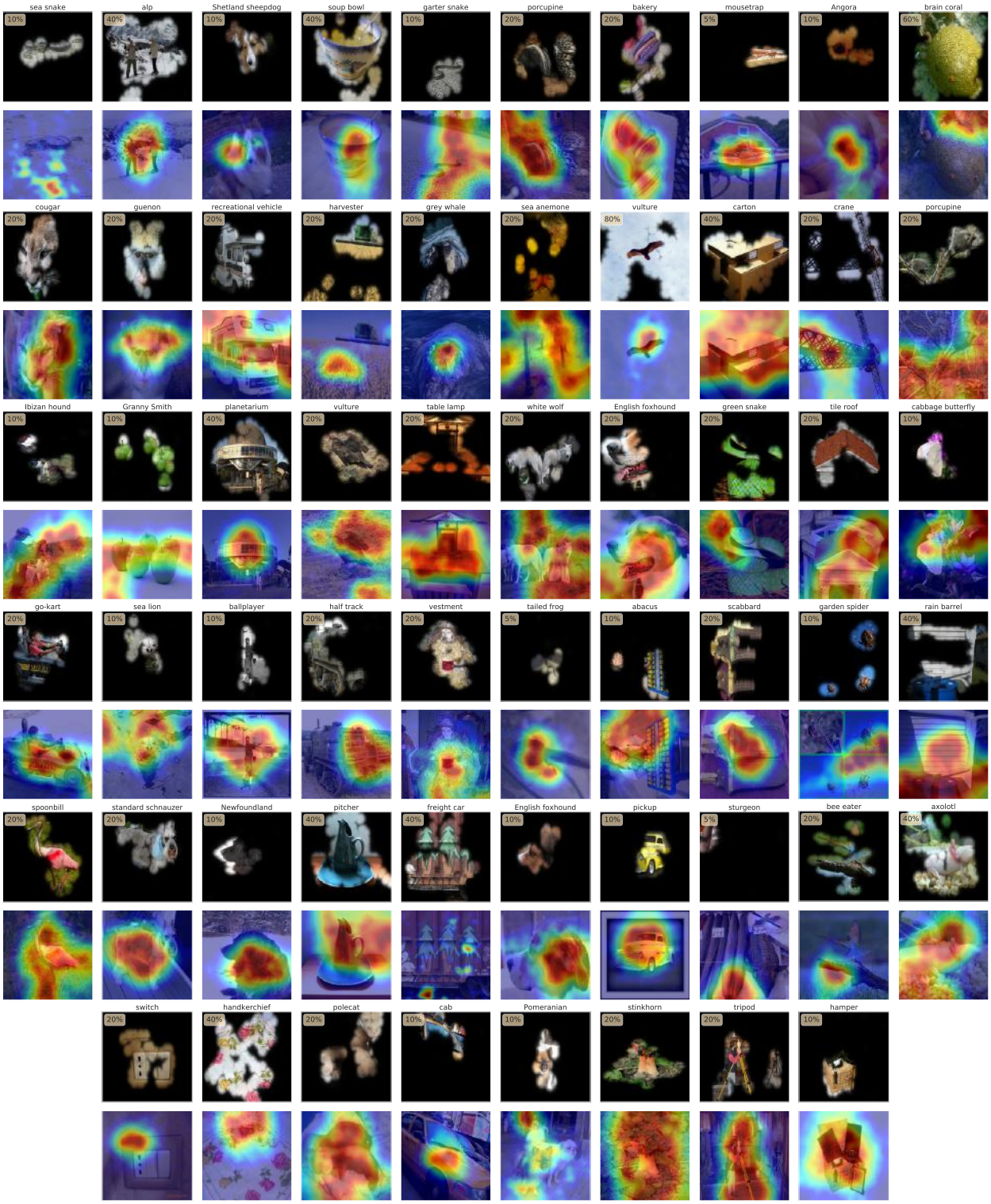

Figure 5: **Comparison with [2]. Odd rows:** Our masks at the optimal area on the first 60 images in the ImageNet validation split. **Even rows:** [2]'s masks. We can see that our masks are able to pick out distinct object instances, whereas [2]'s masks only highlight one connected component due to their smoothness regularization. We also see qualitatively that our method is more stable than [2]'s (where sometimes the mask is completely off the object and/or has an unnatural speckled nature).

Figure 6: **Area growth (up to 5%).**

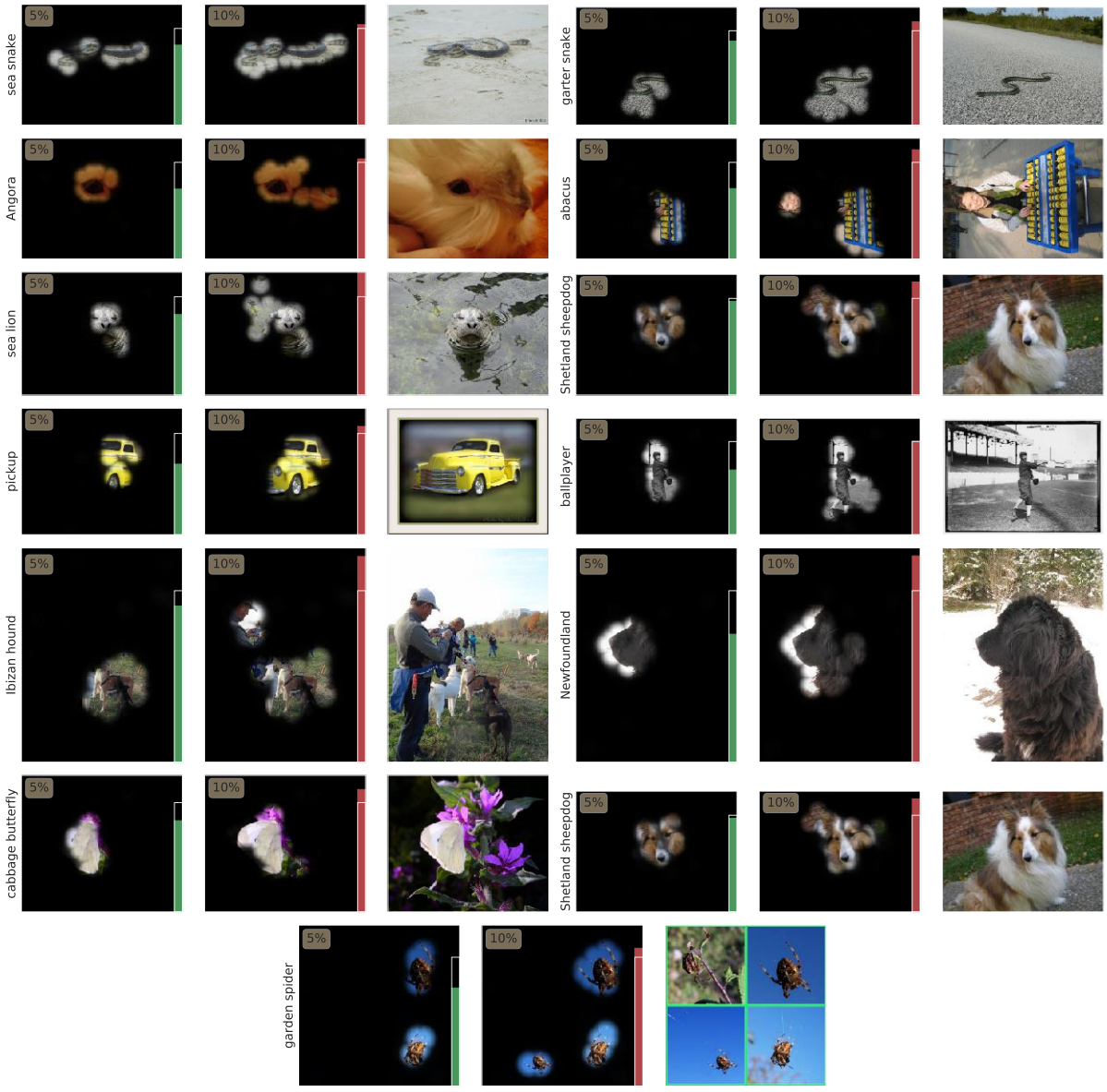

Figure 7: **Area growth (up to 10%).**

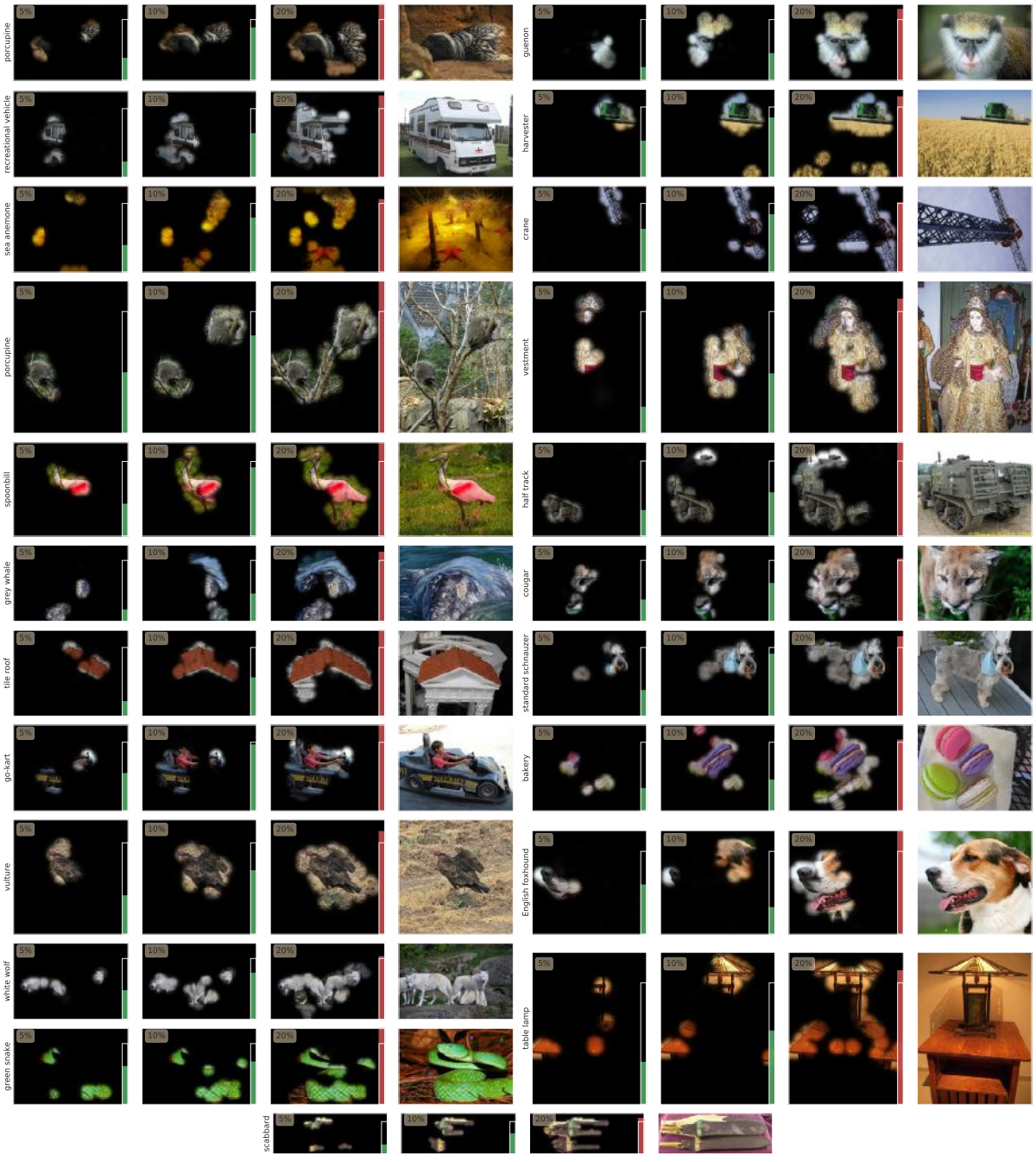

Figure 8: **Area growth (up to 20%).**

Figure 9: **Area growth (up to 40+%).**

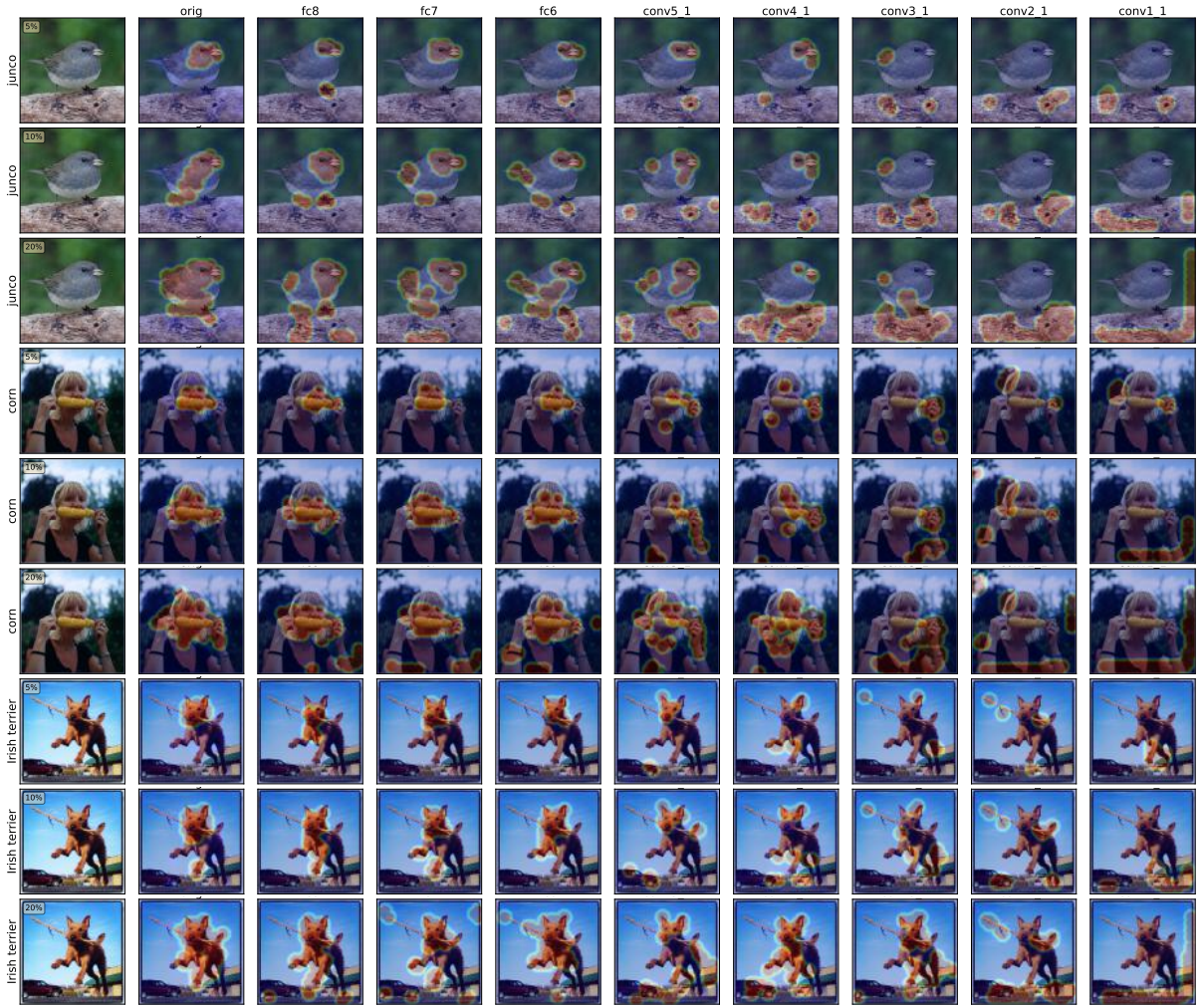

Figure 10: **Cascading randomization sanity check (ours).** Layers with learnable weights are *successively* re-initialized, starting from fc8 to conv1_1. $a^* =$ 20%, 10%, and 5% for "jungo," "corn," and "Irish terrier" respectively. The visualizations look random starting at conv5_1 or conv4_1. Because the maximally activated output neuron is being optimized, when only FC layers are randomized, the output neuron effectively codes for a random rotation on top of conv5_3 (which still encodes object characteristics).

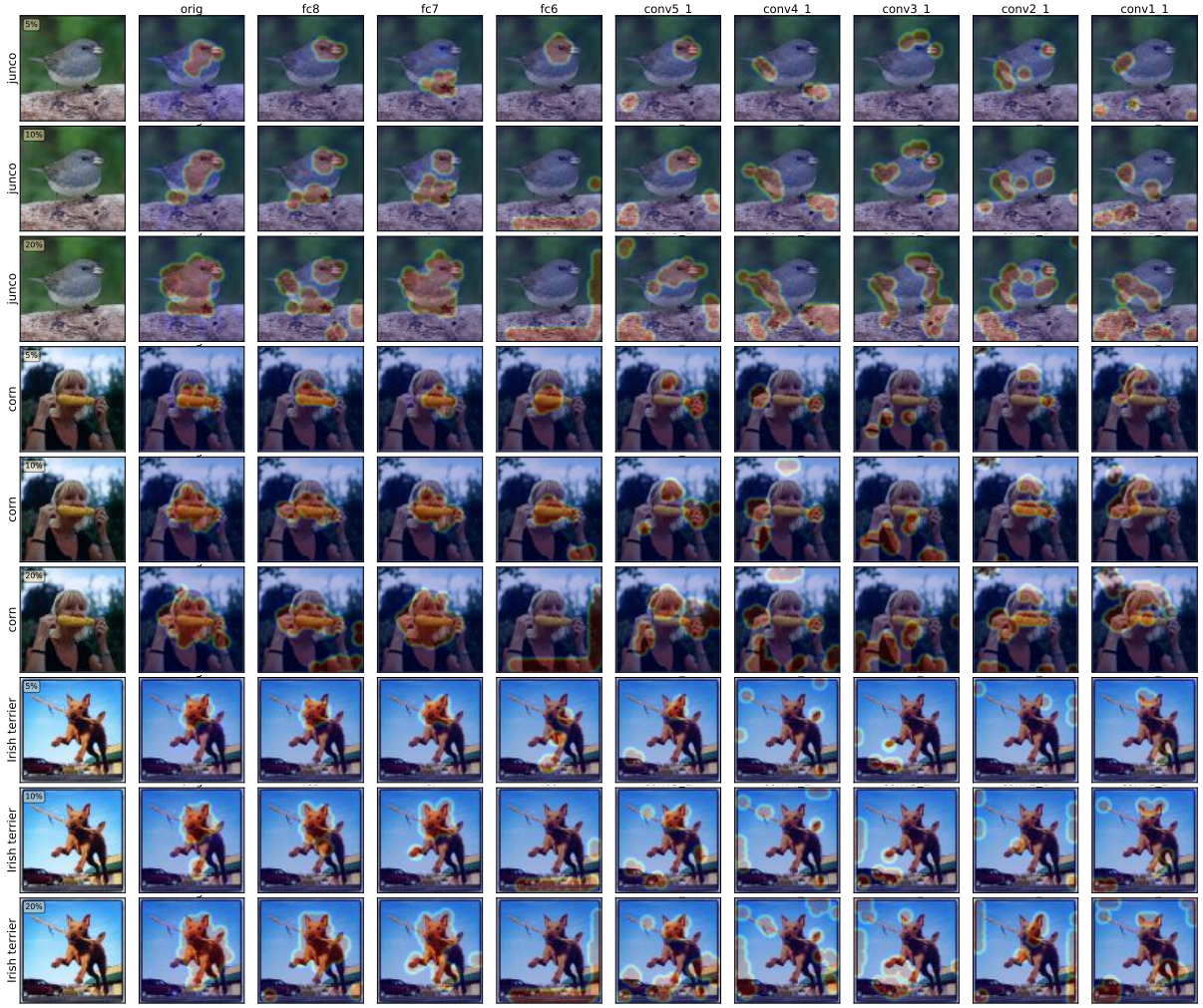

Figure 11: **Independent randomization sanity check (ours).** A single layer's learnable weights are re-initialized (as opposed to progressively randomizing weights as in fig. 10). $a^* = 20\%$, 10%, and 5% for "jungo," "corn," and "Irish terrier" respectively. The visualizations look random around fc6 or conv5_1.

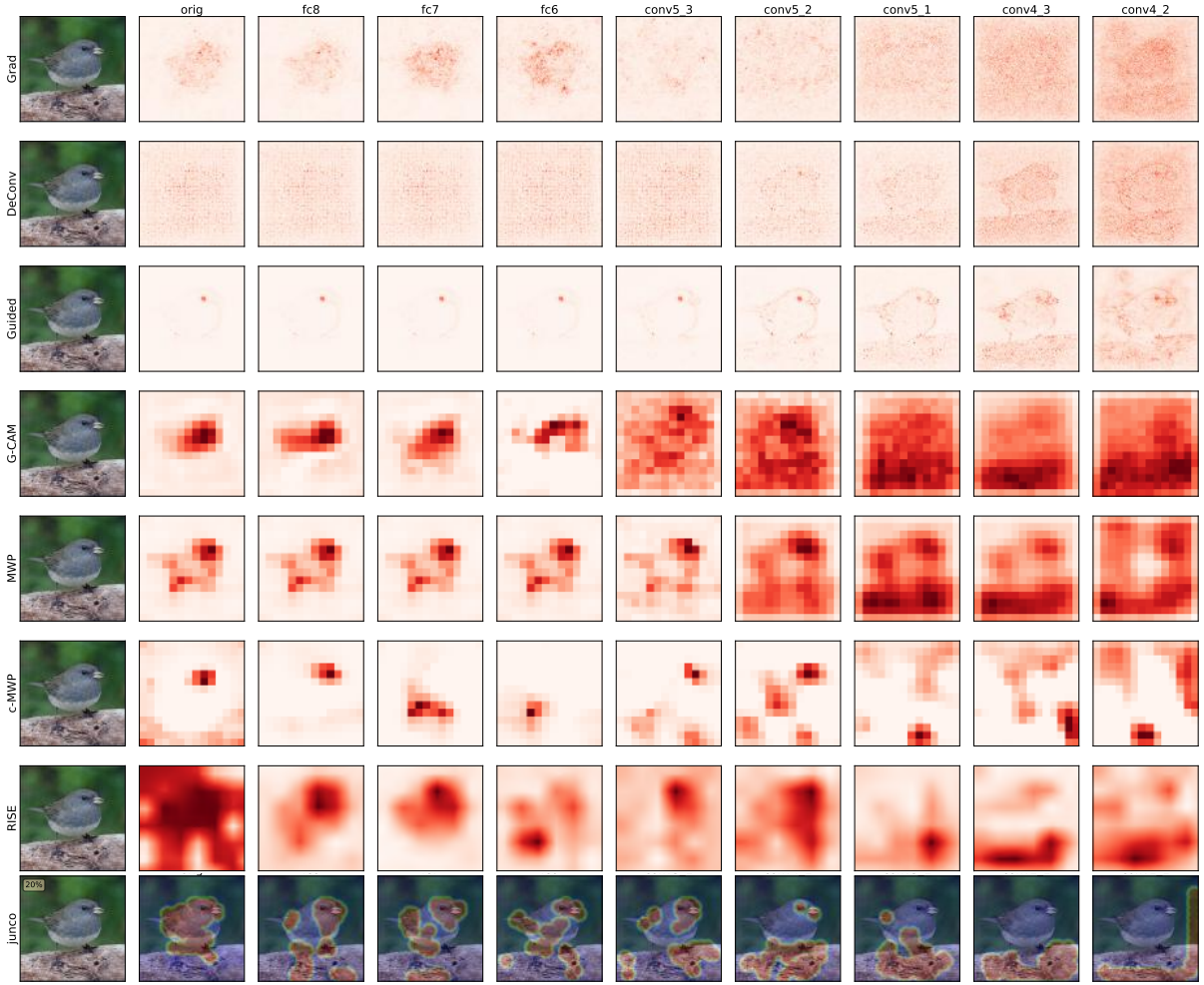

Figure 12: **Cascading randomization sanity check for "jungo" (comparison). Bottom row: our extremal perturbations.**

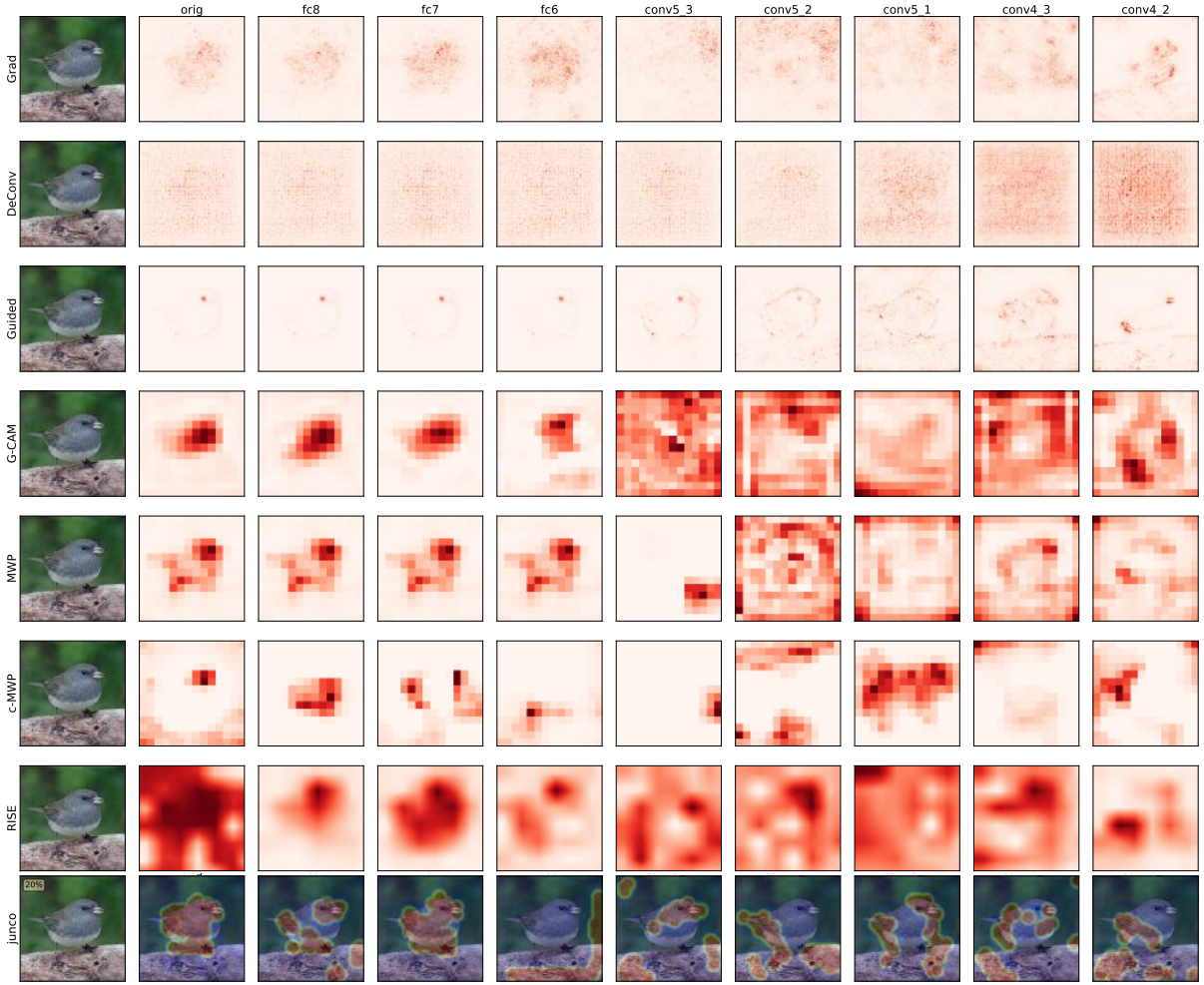

Figure 13: **Independent randomization sanity check for "jungo" (comparison).** Bottom row: our extremal perturbations.

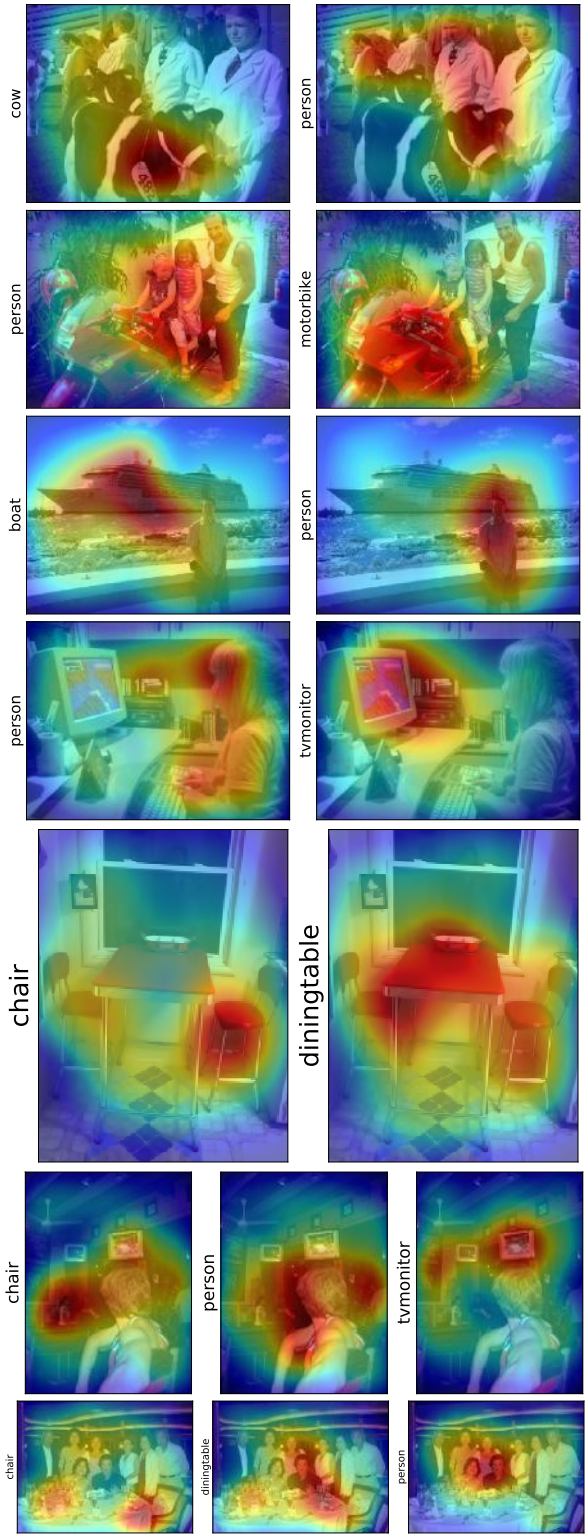

Figure 14: **Pointing Examples from PASCAL**. Here we see a few examples of where our method is able to localize well to different objects in the image. The overlaid visualization is the average mask after Gaussian smoothing ($\sigma = 20$).

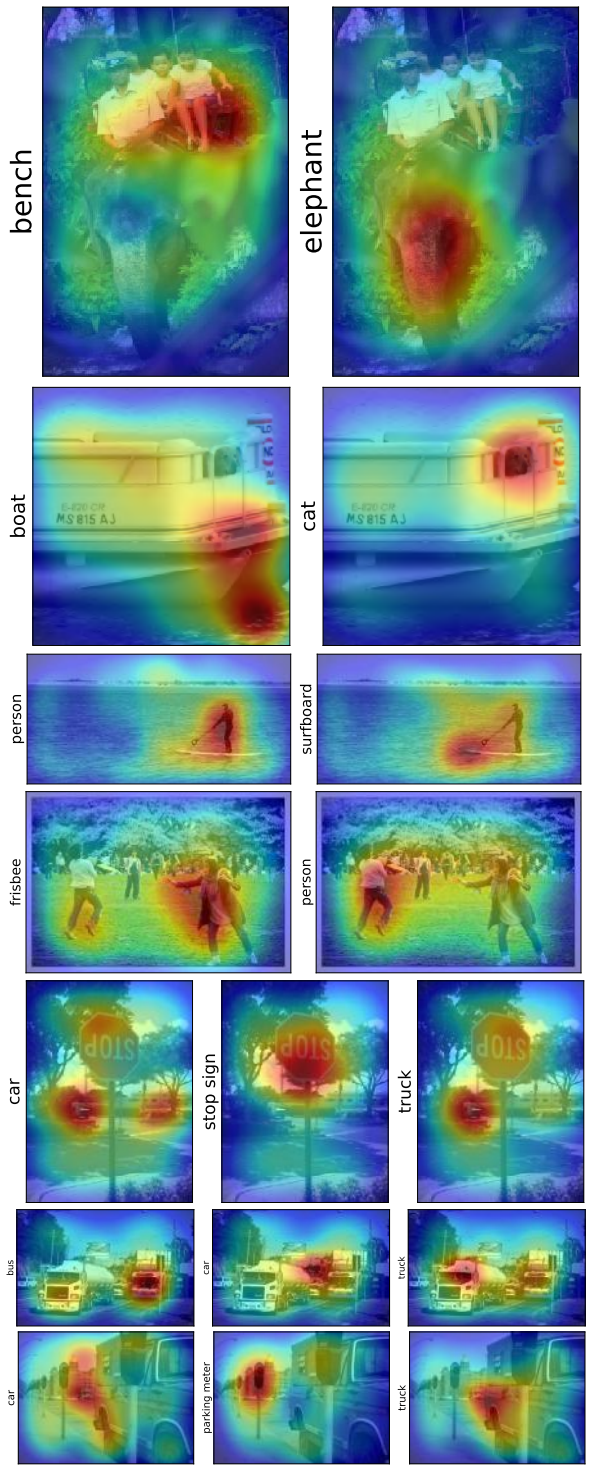

Figure 15: **Pointing Examples from COCO**. The overlaid visualization is the average mask after Gaussian smoothing ($\sigma = 20$).

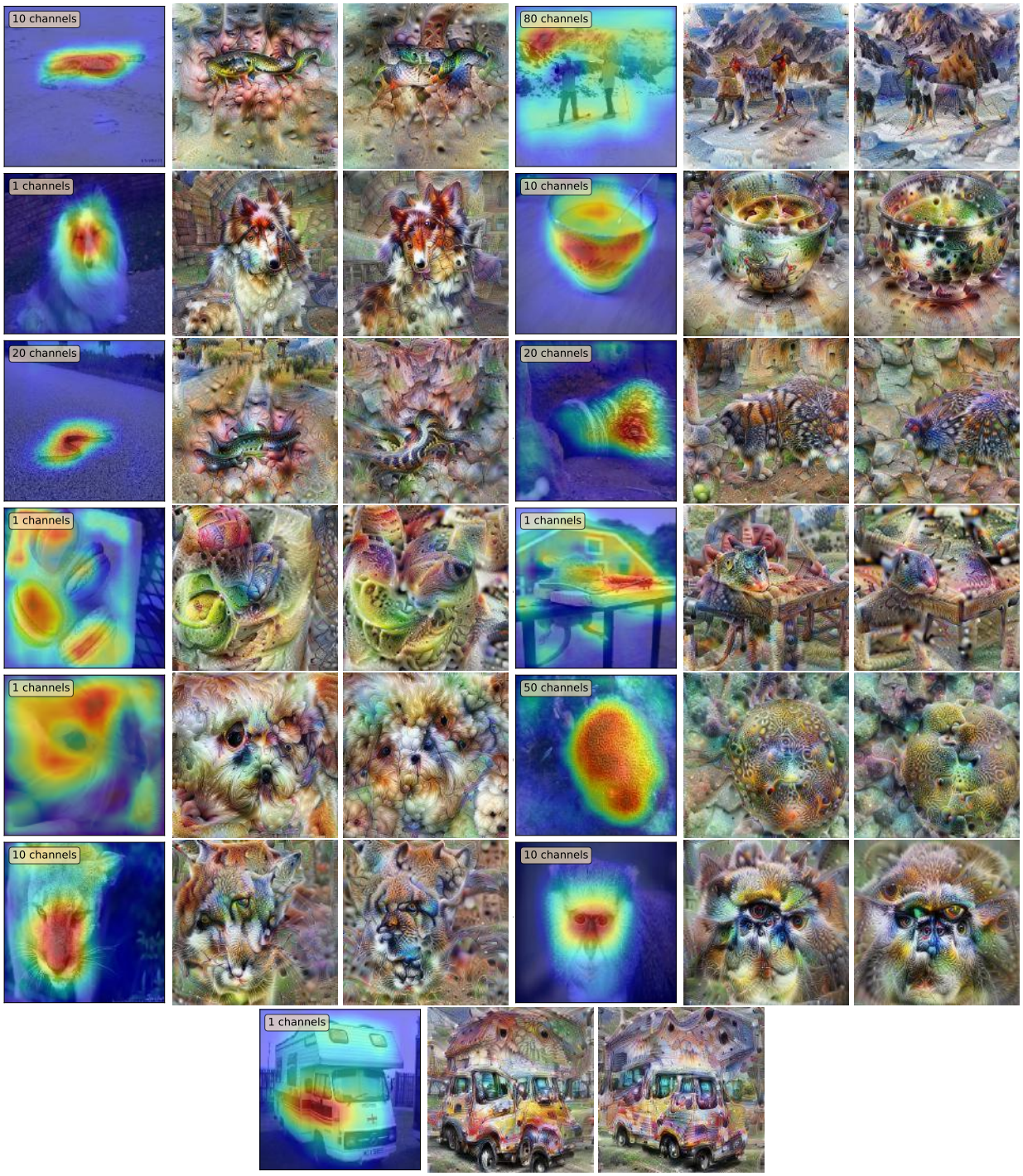

Figure 16: **Per-instance channel attribution visualization (1-14). Left:** input image overlaid with channel saliency map (see eq. 13 in main paper). **Middle:** feature inversion of *original* activation tensor. **Right:** feature inversion of activation tensor *perturbed* by optimal channel mask $\boldsymbol{m}_{a^*}$. By comparing the difference in feature inversions between un-perturbed (middle) and perturbed activations (right), we can identify the salient features that our method highlights. Notable examples are the crib (3rd row, left) and snakes (1st row, left and 3rd row, right). These are visualizations for the ImageNet validation images 1-14.

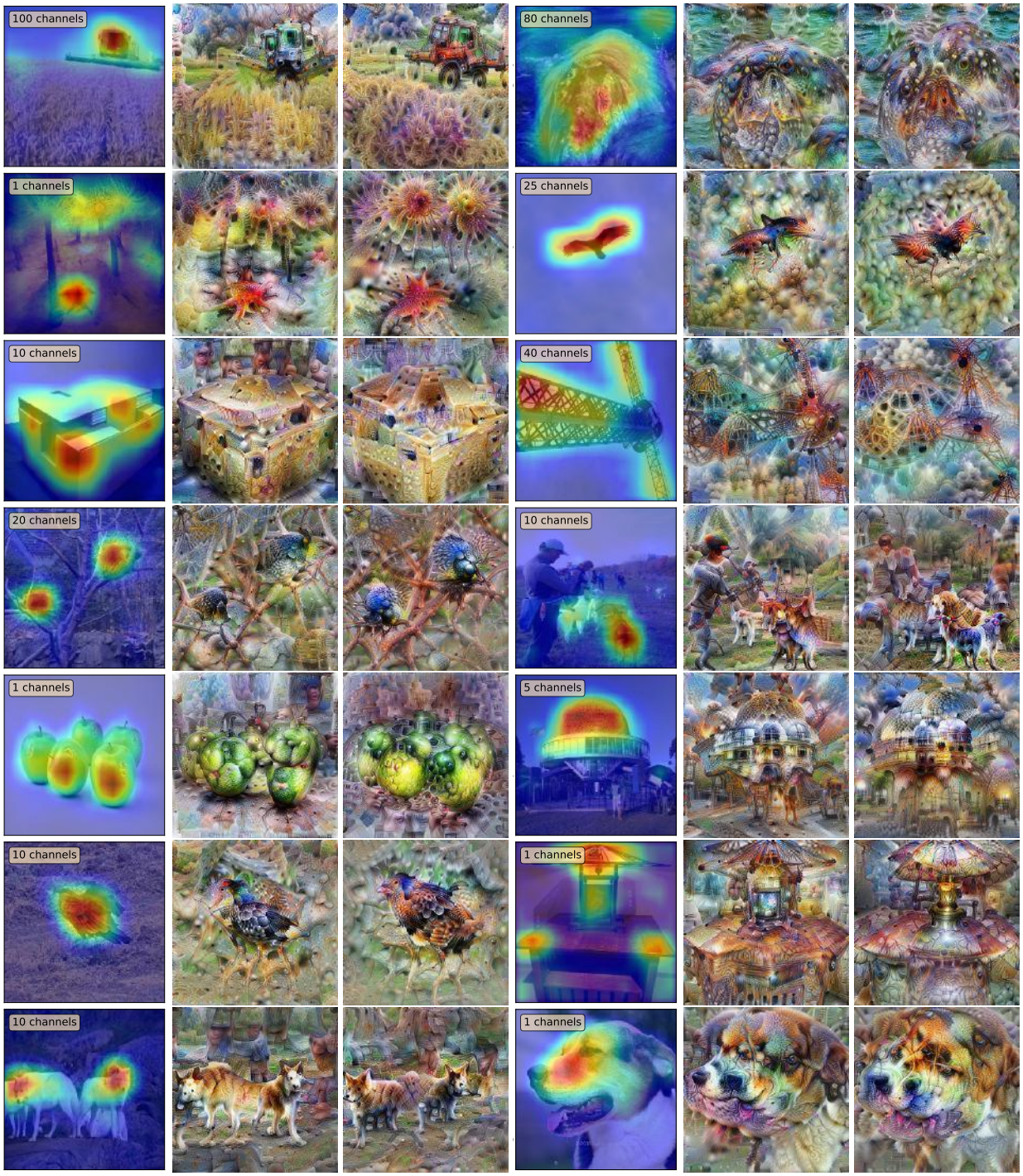

Figure 17: **Per-instance channel attribution visualization (15-28). Left:** input image overlaid with channel saliency map. **Middle:** feature inversion of *original* activation tensor. **Right:** feature inversion of activation tensor *perturbed* by optimal channel mask $\boldsymbol{m}_{a^*}$. Notable examples are the harvester (1st row, left), porcupines (4th row, left), and the bird (2nd to last row, left). These are visualizations for the ImageNet validation images 15-28.

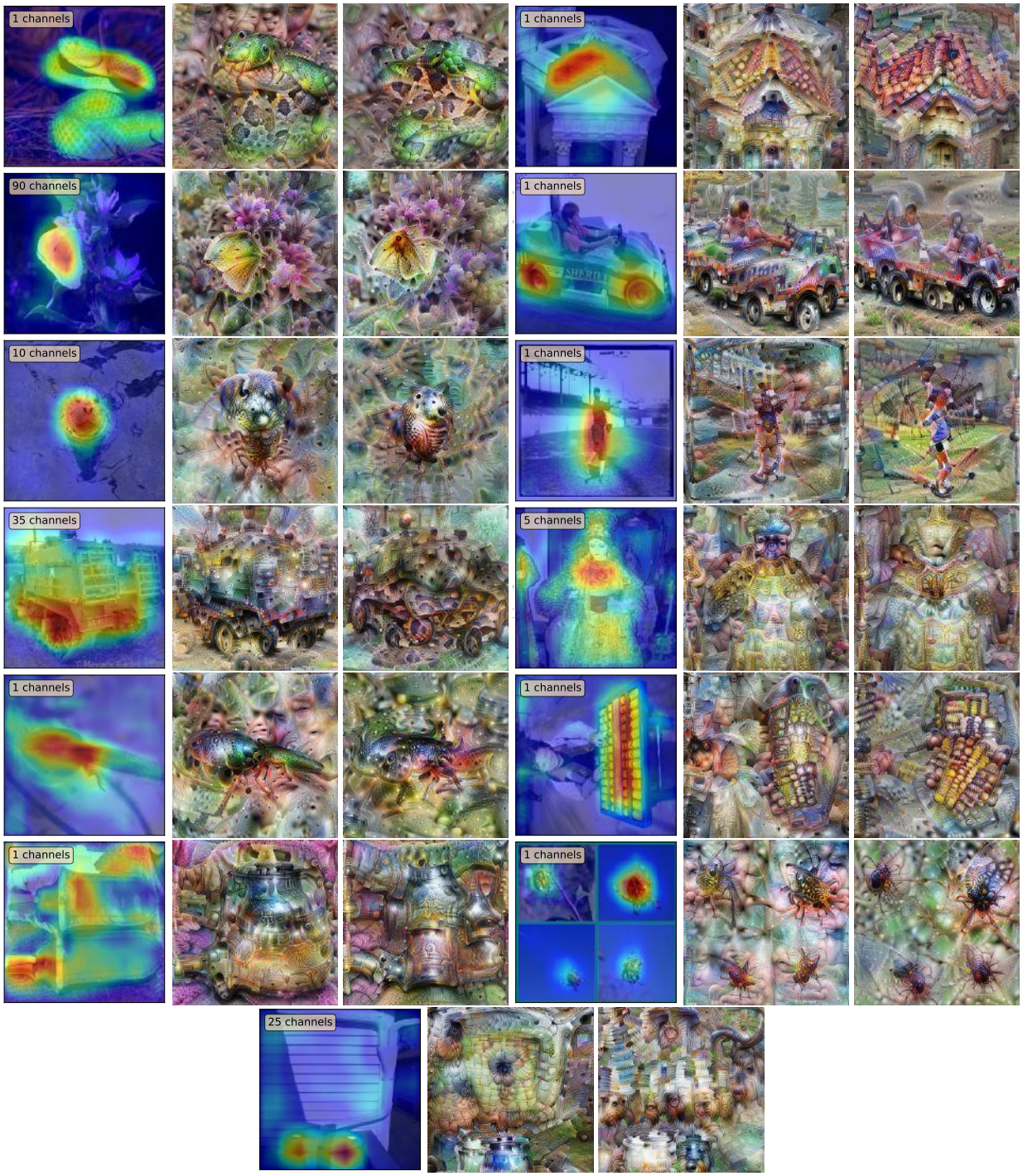

Figure 18: **Per-instance channel attribution visualization (29-42). Left:** input image overlaid with channel saliency map. **Middle:** feature inversion of *original* activation tensor. **Right:** feature inversion of activation tensor *perturbed* by optimal channel mask $\boldsymbol{m}_{a^*}$. Notable examples are the baseball player (3rd row, right), abacus (3rd to last row, right), crib (last row, left), and rain barrel (last row, right). These are visualizations for the ImageNet validation images 29-42.