A. Architecture

For the **feature extractor** F, we use the *ResNet50 v2* architecture (He et al., 2016; Kolesnikov et al., 2019) with the standard channel widening factor of 4 (i.e. 16×4 channels in the first convolutional layer) and a representation size of 2048 at the *pre-logits* layer unless otherwise noted.

For the **lens** L, we use a variant of the *U-Net* architecture (Figure 11; Ronneberger et al. 2015). The lens consists of a convolutional encoder and decoder. The encoder and decoder are each a stack of n residual units (same unit architecture as used for the feature extractor), with k channels for the first unit of the encoder. We use $n = 4$ and $k = 64$ for all experiments. Two additional residual units form the bottleneck between encoder and decoder (see Figure 11). After each unit in the encoder, the number of channels is doubled and the resolution is halved by max-pooling with a 2×2 kernel and stride 2. Conversely, after each decoder unit, the number of channels is halved and the resolution is doubled using bilinear interpolation. At each resolution level, skip connections are created between the encoder and decoder by concatenating the encoder representation channel-wise with the decoder representation before applying the next decoder unit. The output of the decoder is of the same resolution as the input image, and reduced to three channels by a 1×1 convolutional layer. This map is combined by element-wise addition with the input image to produce the lens output.

We choose the *U-Net* architecture because it efficiently combines a large receptive field with a high output resolution. For example, for input images of size 224×224, the maps at the bottleneck of the *U-Net* are of size 14×14, such that a 3×3 convolution at that size corresponds to 48×48 pixels at the input resolution and is able to capture large-scale image context. Furthermore, the skip connections of the *U-Net* make it trivial for the lens to reconstruct the input image by setting all internal weights to zero. This is important to ensure that the changes made by the lens to the image are not simply due to a lack of capacity.

We find that a lens with $n = 4$ and $k = 64$ yields good results in general, although initial experiments suggested that tuning the lens capacity individually for each pretext task and dataset may provide further gains.

We also tested how the performance of the lens varies with the capacity of the feature extraction network. For the *Rotation* task and *ImageNet*, we trained models with different widening factors (channel number multiplier). As expected, wider networks perform better (Figure 10). We find that the lens improves accuracy across all model widths. The accuracy gain of applying the lens to a feature extraction network with a width factor of 4 is equivalent to the gain obtained by widening the network by 2–4×.

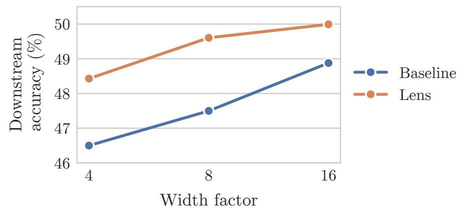


Figure 10. Downstream accuracy for *Rotation* models trained on *ImageNet* with different feature extraction network widening factors. The performance gain remains large across model sizes.

For the experiments using *CIFAR-10* (Figure 3), we used a smaller lens architecture consisting of a stack of five ResNet50 v2 residual units without down or up-sampling.

B. Downstream evaluation

For downstream evaluation of learned representations, we follow the linear evaluation protocol with SGD from Kolesnikov et al. (2019). A logistic regression model for *ImageNet* or *Places205* classification was trained using SGD on the representations obtained from the pre-trained self-supervised models.

For training the logistic regression, we preprocessed input images in the same way for all models: Images were resized to 256×256, randomly cropped to 224×224, and the color values were scaled to $[-1, 1]$. For evaluation, the random crop was replaced by a central crop.

Representations were then obtained by passing the images through the pre-trained models and extracting the *pre-logits* activations. For patch-based models, we obtained representations of the full image by averaging the representations of nine patches created from the full image. To create the patches, the the central 192×192 section of the 224×224 input image was divided into a 3×3 grid of patches. Each patch was passed through the feature extraction network and the representations were averaged.

The logistic regression model was trained with a batch size of 2048 and an initial learning rate of 0.8. We trained for 90 epochs and reduced the learning rate by a factor of 10 after epoch 50 and epoch 70. For both *ImageNet* and *Places205*, training was performed on the full training set and the performance is reported for the public validation set.

C. Adversarial training with FGSM

For the comparison to adversarial training (Table 1), we used the fast gradient-sign method (FGSM) as described by

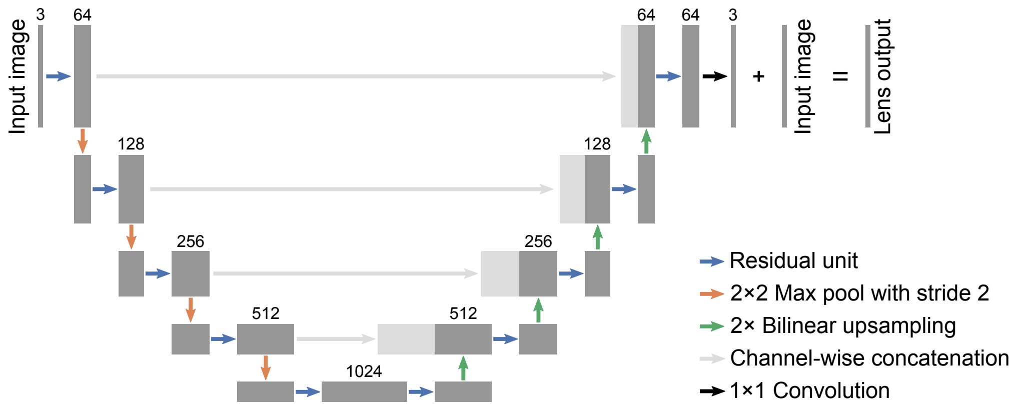


Figure 11. Lens architecture. The number of channels is indicated above each block. Based on (Ronneberger et al., 2015).

Table 2. Evaluation of representations from models trained on *ImageNet* with different self-supervised pretext tasks, using lensed-image representations only, without concatenating non-lensed representations. Otherwise like Table 1: The scores are accuracies (in %) of a logistic regression model trained on representations obtained from the frozen models. Mean ± s.e.m over three random initializations. Values in bold are better than the next-best method at a significance level of 0.05. Training images are preprocessed as suggested by the respective original works.

Dataset	Method	Pretext task			
		Rotation	Exemplar	Rel. patch loc.	Jigsaw
ImageNet	Baseline	45.9 ± 0.04	42.2 ± 0.27	37.5 ± 0.17	34.6 ± 0.10
	Lens	46.9 ± 0.09 $(+\mathbf{1.06})$	44.5 ± 0.12 $(+\mathbf{2.26})$	39.1 ± 0.13 $(+\mathbf{1.63})$	38.2 ± 0.09 $(+\mathbf{3.63})$
Places205	Baseline	41.3 ± 0.13	41.8 ± 0.15	40.2 ± 0.09	38.8 ± 0.21
	Lens	41.8 ± 0.14 $(+\mathbf{0.53})$	42.4 ± 0.20 $(+0.60)$	40.9 ± 0.05 $(+\mathbf{0.70})$	40.5 ± 0.11 $(+\mathbf{1.74})$

Kurakin et al. (2016). Analogously to our sweeps over λ for the lens models, we swept over the perturbation scale $\epsilon \in \{0.01, 0.02, 0.04, 0.08, 0.16\}$ and report the accuracy for the best ϵ in Table 1. As suggested by Kurakin et al. (2016), we randomized the perturbation scale for each image by using the absolute value of a sample from a truncated normal distribution with mean 0 and standard deviation ϵ. Since this randomization already includes nearly unprocessed images, we do not include further unprocessed images during training.

D. Case study: SimCLR

Concurrently with our work, a powerful new self-supervised approach based on contrastive learning, called *SimCLR*, was published (Chen et al., 2020). Here, we describe our experience applying automatic shortcut removal to *SimCLR* as an informal "case study". Our goal is to provide a practical example for how our method can be applied to understand and improve new self-supervised tasks. Even though we find that the lens does not improve the linear evaluation performance of *SimCLR*, the lens provided insights that allowed us to improve SimCLR performance on other tasks.

D.1. Linear evaluation on *ImageNet*

As a first step, we applied automatic shortcut removal as described in the main paper to *SimCLR*[4] and evaluated the learned representations with the linear protocol. As we suggest in the main paper, we ran a sweep across the reconstruction loss scale λ and left the other hyperparameters at their default values. Figure 12 shows that applying the lens to *SimCLR* does not improve representation quality under the linear evaluation protocol. The performance increases monotonically with λ and always remains below the baseline performance of 68.90 % (*ResNet50x1*), suggesting that any amount of lens-induced perturbation is harmful for this task under the linear evaluation protocol. To understand this result, we turned to inspecting the lens outputs.

D.2. Lens outputs

The lens outputs (Figure 13) indicate that the lens primarily reduces color saturation and causes blurring of high-frequency image components. This suggests that the lens attacks features in a way that is similar to the augmentations that are part of the standard *SimCLR* code, specifically

[4]Code available at https://github.com/google-research/simclr; we used SimCLRv1.

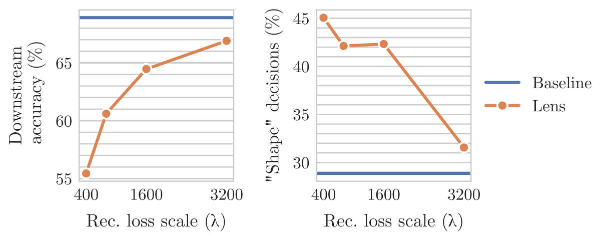


Figure 12. **Left:** Linear evaluation performance of *SimCLR* on *ImageNet*. **Right:** Fraction of "shape" decisions on the conflict stimuli from Geirhos et al. (2019)

Gaussian blur and *Color jitter*. These augmentations are an integral part of *SimCLR*. We hypothesize that the augmentations were already so highly optimized that any additional image perturbation leads to a decrease in performance. Consistently, in separate experiments, we found that if we ablate the *Gaussian blur* and *Color jitter* augmentations, applying the lens improves over the ablated baseline (but not beyond the un-ablated baseline performance). While the lens does not provide further improvements on top of the hand-designed augmentations, it is encouraging that the lens identifies the same perturbations that were chosen by the expert authors of *SimCLR*.

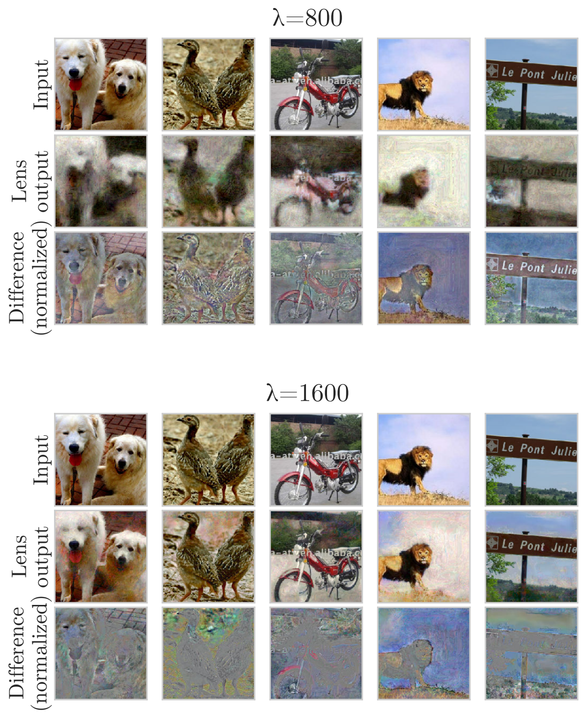


Figure 13. Example lens outputs for *SimCLR*.

Table 3. Fine-tuning performance of *SimCLR R50x1* on the *Visual Task Adaptation Benchmark* (Zhai et al., 2019). Abbreviations: Spec., Specialized; Struct., Structured.

	mean	Natural	Spec.	Struct.
Baseline	48.79	51.08	74.74	33.82
Lens	51.51 (**+2.72**)	49.96	76.89	40.17

D.3. Semanticity

The lens output suggests that high-frequency patterns, as well as colors, are important shortcut features for *SimCLR*. We therefore hypothesized that the representations learned by *SimCLR* primarily encode texture details, rather than high-level shape information. Indeed, evaluating *SimCLR* on the dataset from Geirhos et al. (2019) as in Section 4.2.5, showed that *SimCLR* makes shape-based decisions in only 28.86% of cases (Figure 12). Applying the lens to *SimCLR* increases the proportion of shape-based decisions to over 40%, which indicates that the lens strongly shifts the network towards more semantic representations.

D.4. Improvements on other tasks

While it has been shown that natural image classification tasks such as *ImageNet* classification can be solved accurately based on texture information (Geirhos et al., 2019), other tasks might benefit from the additional semantic information that is learned when the lens is used. To investigate this question, we turned to the *Visual Task Adaptation Benchmark* (*VTAB*, Zhai et al. 2019), which is a collection of 19 tasks that span *natural*, *specialized* and *structured* domains. Indeed, we find that automatic shortcut removal improves the mean score of *SimCLR* on *VTAB* by 2.72% (Table 3). This improvement comes primarily from the Specialized and Structured datasets, while the score on Natural datasets is slightly reduced. These results suggest that *SimCLR* representations are highly adapted to *ImageNet*, and their performance on a wider variety of tasks may suffer from shortcuts that can be mitigated with our method.

D.5. Summary

The *SimCLR* case study shows how our method can be used to understand and improve a new pretext task. While our method does not always result in a quick win on all benchmarks, it provides a deeper understanding of the task-specific shortcut features, which may guide the practitioner towards opportunities for improvement.

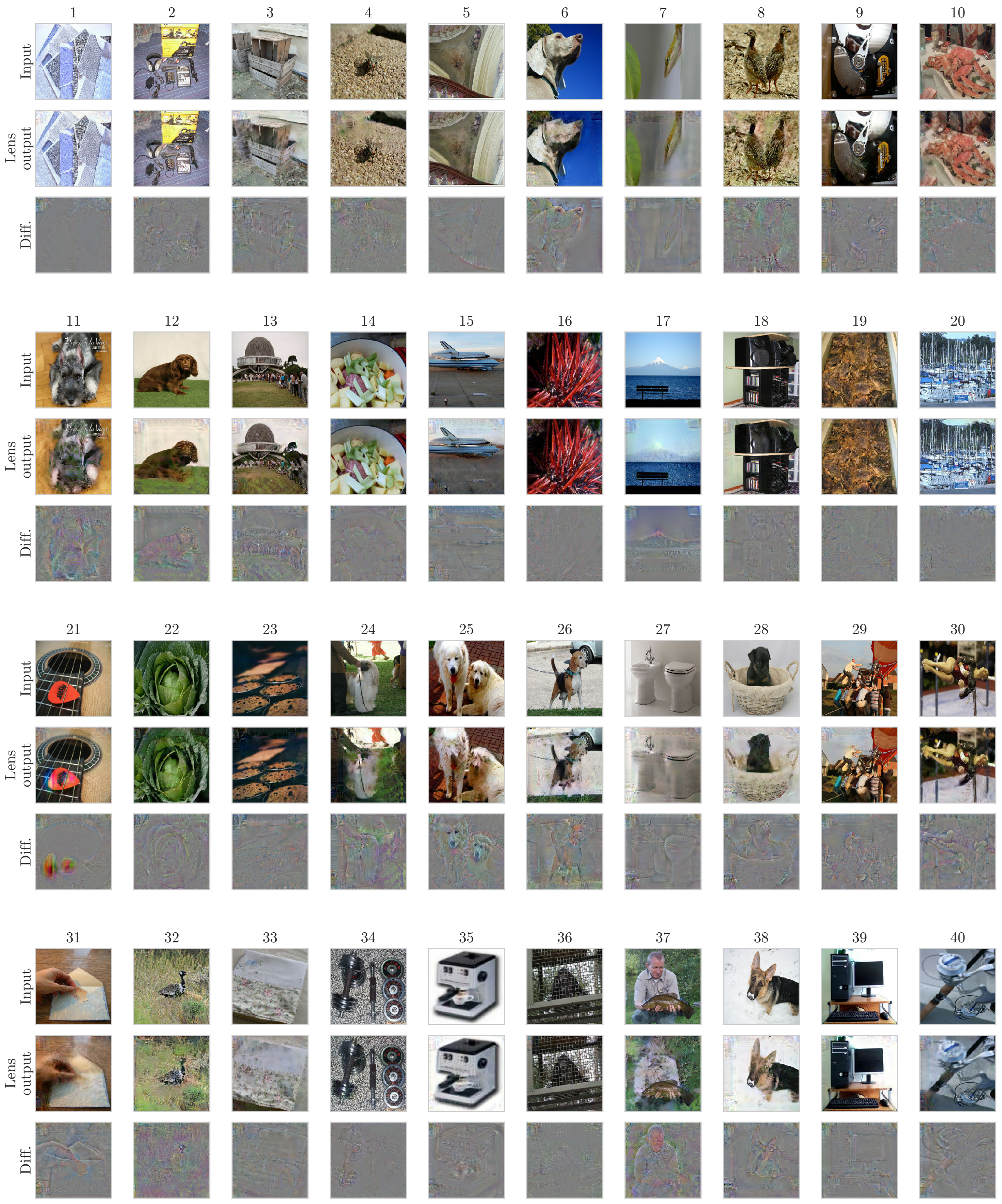


Figure 14. Further example lens outputs for models trained on *ImageNet* with the *Rotation* task. Images were randomly sampled from the *ImageNet* validation set.

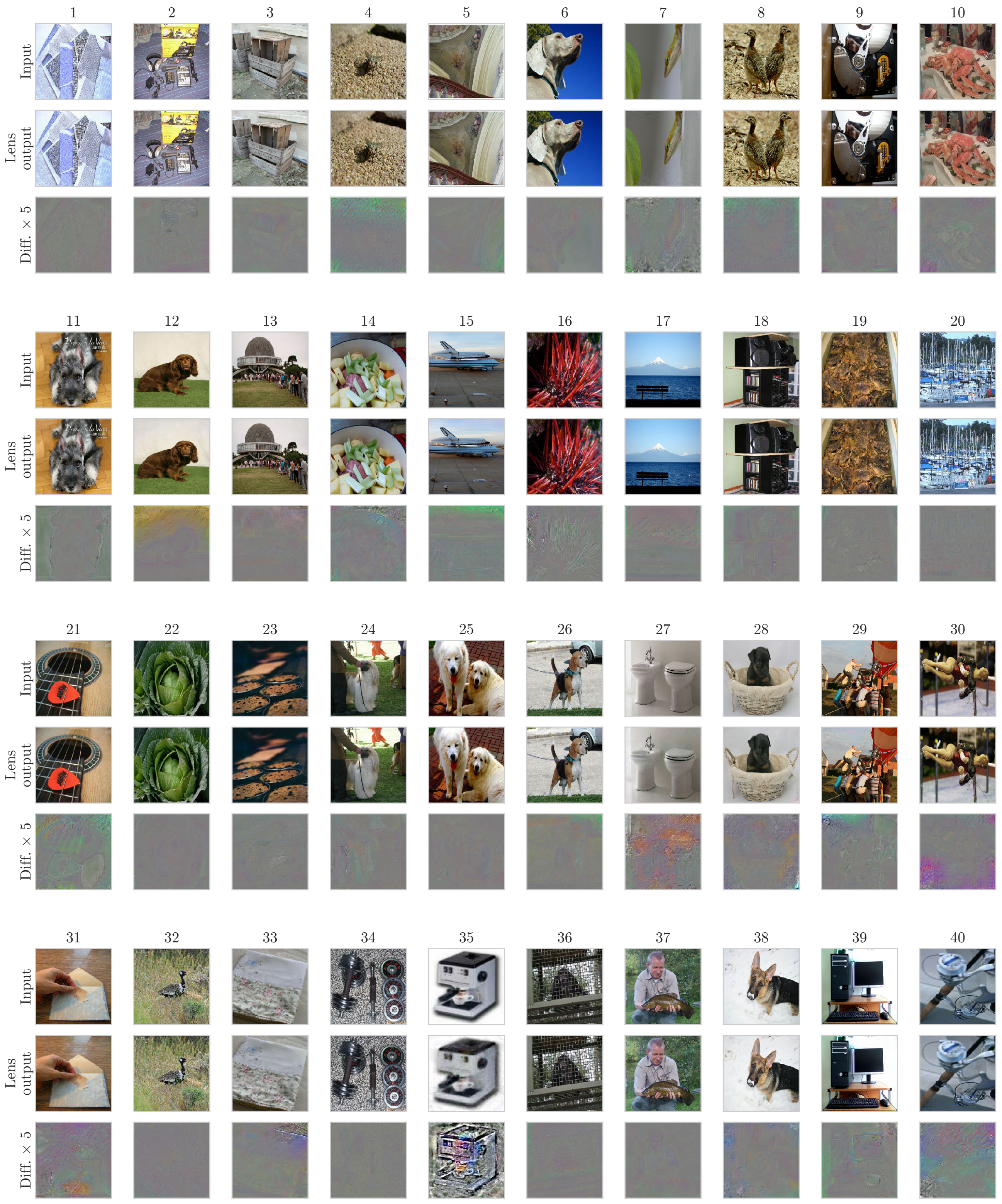


Figure 15. Further example lens outputs for models trained on *ImageNet* with the *Exemplar* task. Images were randomly sampled from the *ImageNet* validation set.

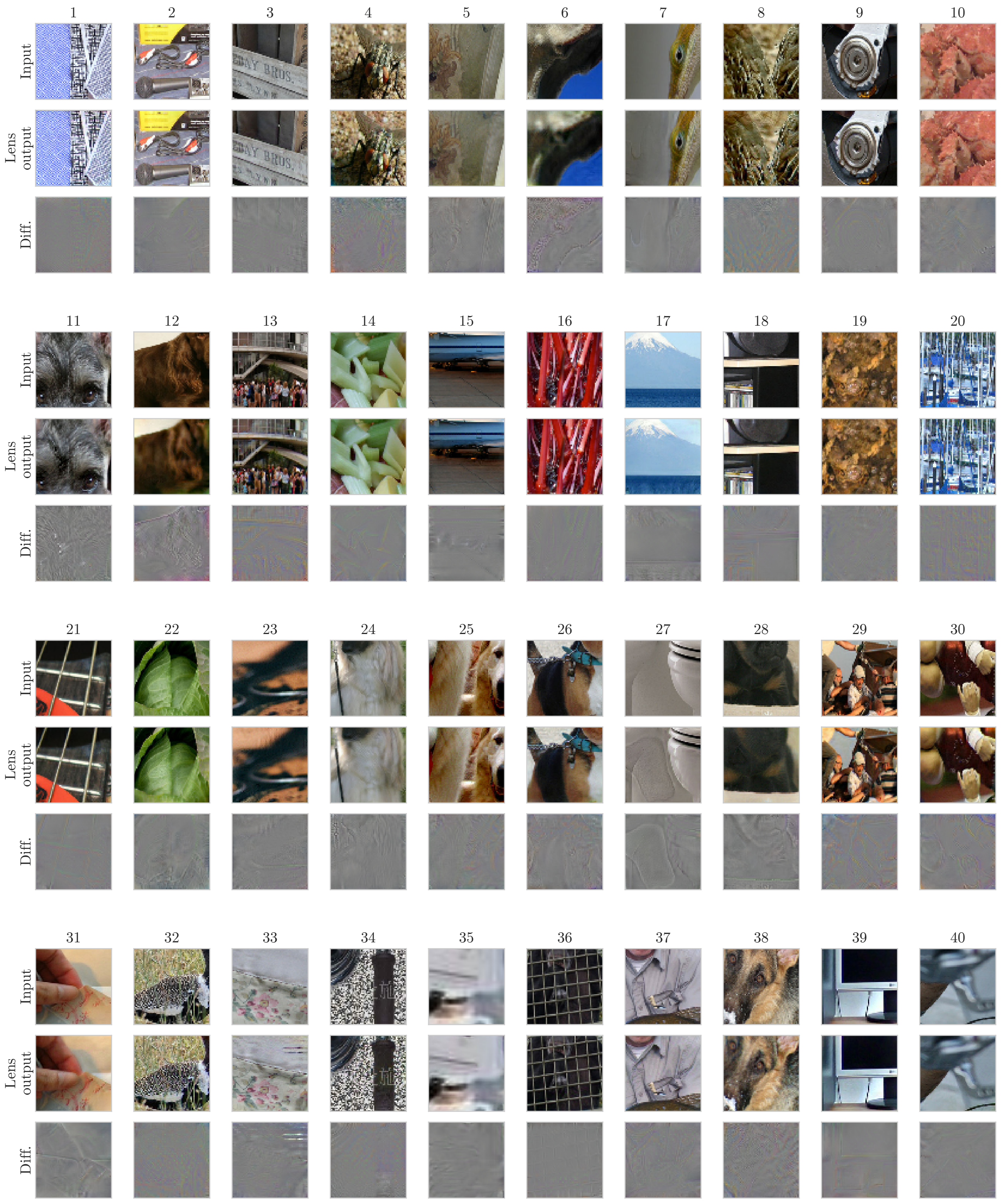


Figure 16. Further example lens outputs for models trained on *ImageNet* with the *Relative patch location* task. Images were randomly sampled from the *ImageNet* validation set.

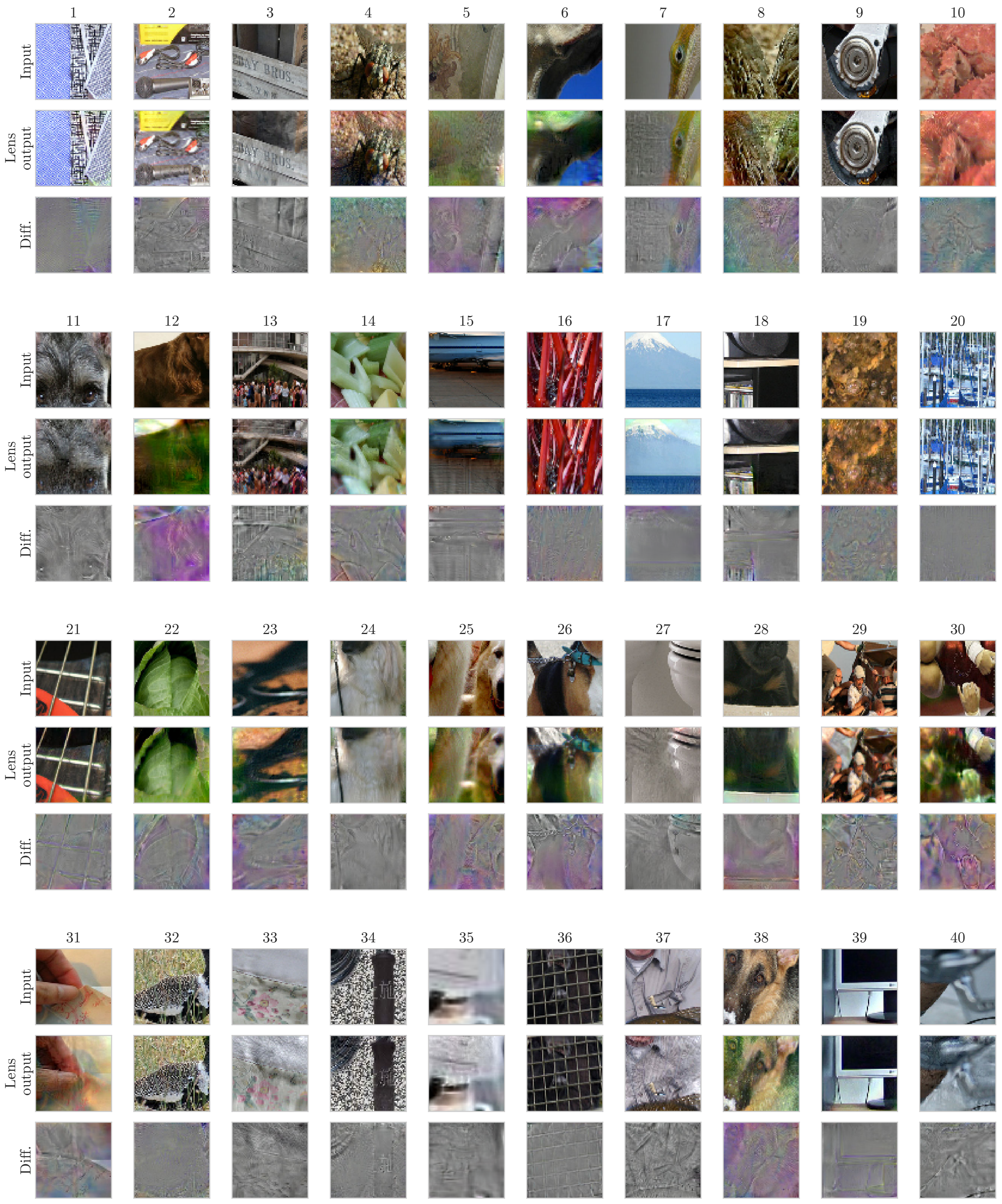
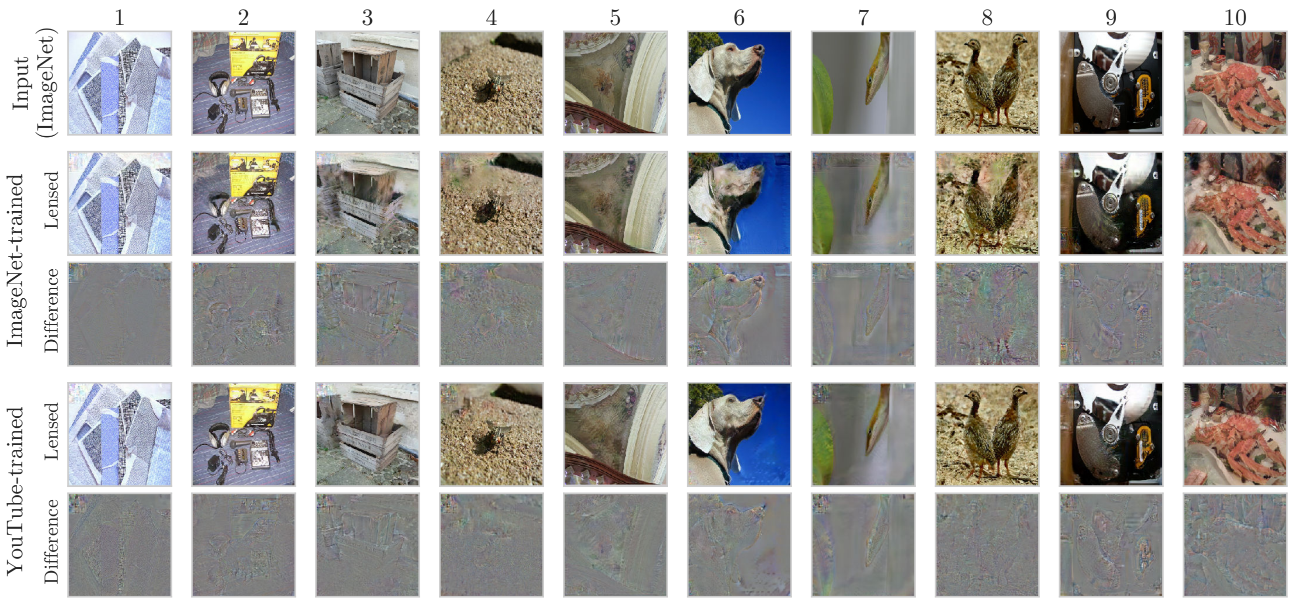


Figure 17. Further example lens outputs for models trained on *ImageNet* with the *Jigsaw* task. Images were randomly sampled from the *ImageNet* validation set.



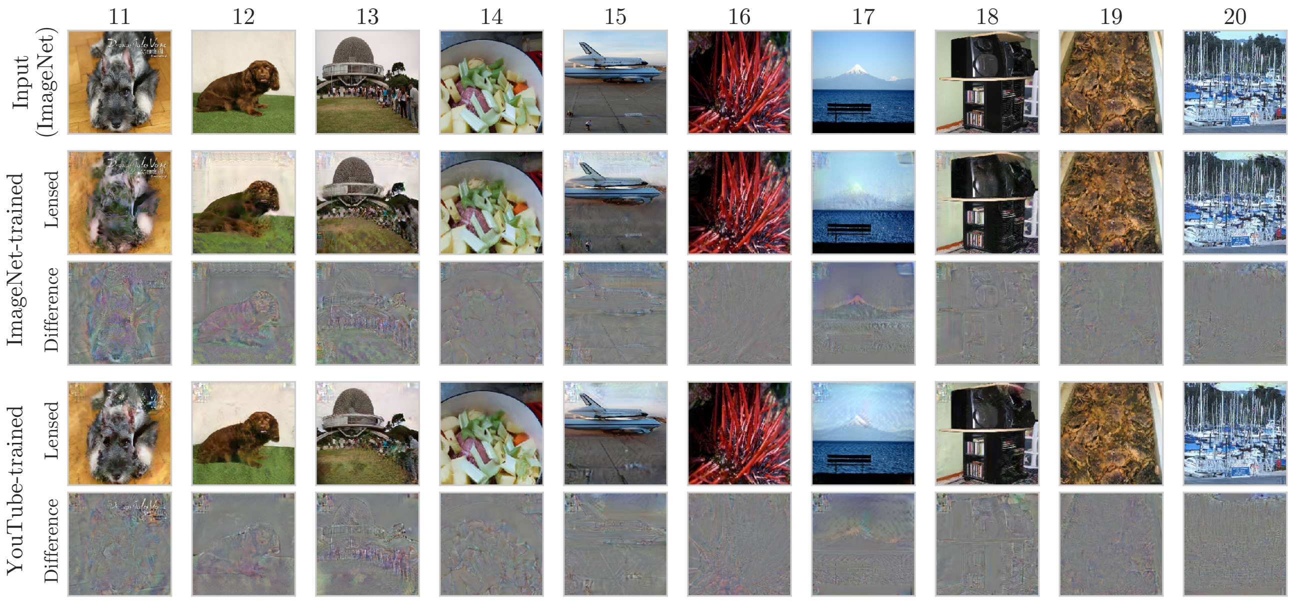


Figure 18. Further example lens outputs for models trained on *YouTube1M* with the *Rotation* task. Outputs from *ImageNet*-trained models are provided for comparison. Images were randomly sampled from the *ImageNet* validation set.

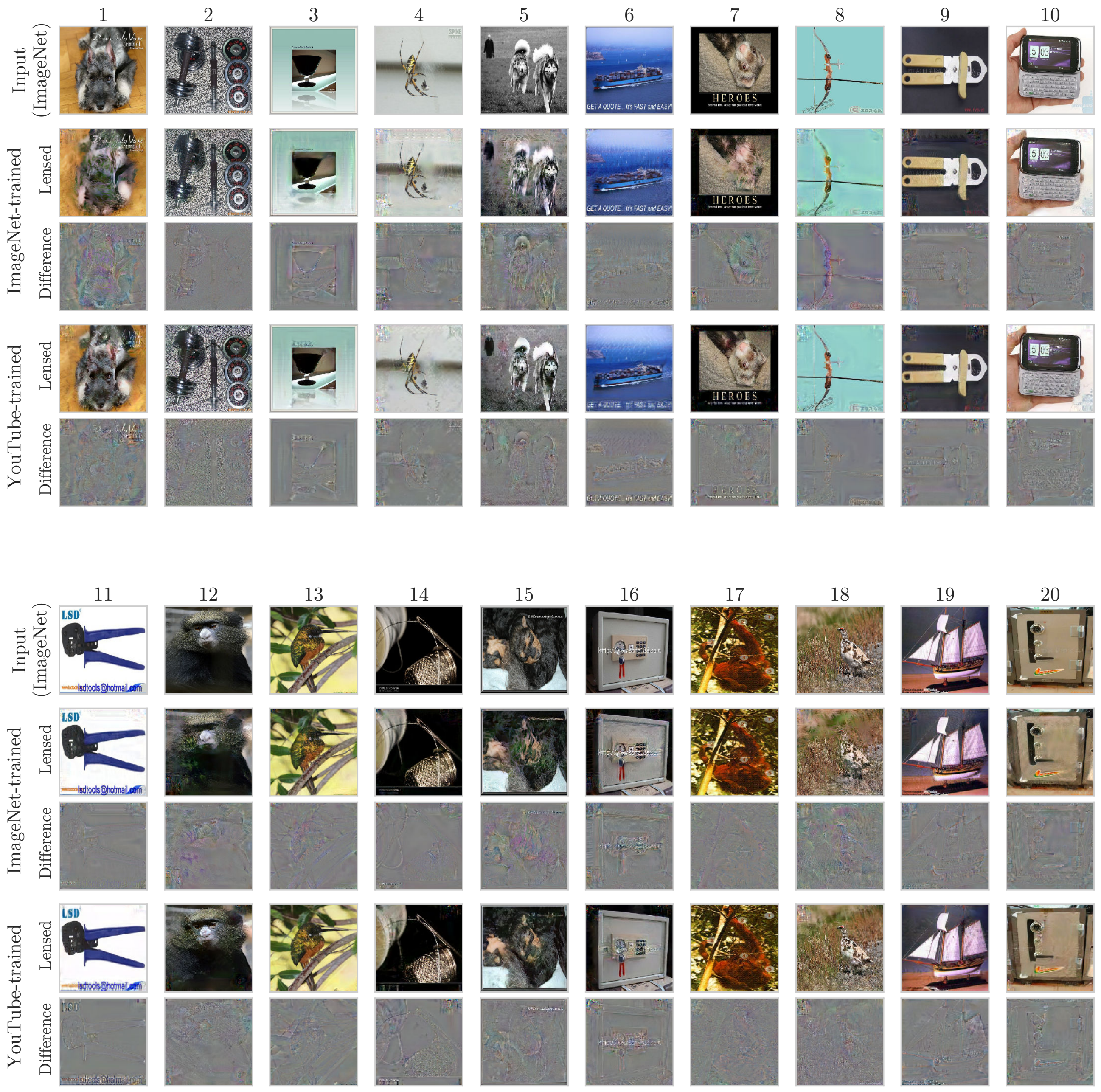


Figure 19. Further example lens outputs for images containing text, comparing models trained on *YouTube1M* and *ImageNet* with the *Rotation* task. Images containing artificially overlaid text (logos, watermarks, etc.) were manually selected from a random sample of 1000 *ImageNet* validation images, before inspecting lens outputs. A random sample of these images is shown.