CASTing Your Model:
Learning to Localize Improves Self-Supervised Representations
(Supplementary)

Ramprasaath R. Selvaraju[1*] Karan Desai[2*] Justin Johnson[2] Nikhil Naik[1]
[1]Salesforce Research, [2]University of Michigan
{rselvaraju,nnaik}@salesforce.com {kdexd,justincj}@umich.edu

Appendices

A. Introduction

This supplementary material is organized as follows. We first provide pseudo-code for our approach, Contrastive Attention-Supervised Tuning (CAST), showing how it can be easily applied on top of any contrastive self-supervised approaches. We then provide a sample of the kinds of random crops that are generated by our saliency-constrained random cropping approach and compare them to regular random crops used by self-supervised learning methods. We then provide randomly sampled qualitative examples that show that CAST-trained feature representations obtain better grounding, both on their own and when finetuned on downstream tasks, such as image classification. We then show qualitative examples from the Background Challenge evaluation reported in Section 4.3 of the main paper.

B. Random Crops

As shown in Fig 1 of the main paper, when training with complex scene-level images, models often receive a noisy training signal—random crops from an image may contain different objects, or none at all. To fix this problem, we design a random crop transform that generates input crops constrained to overlap with the saliency map. In Figure S1, we show how our Saliency-Constrained Random Crops differ from unconstrained random crops used in MoCo and other contrastive approaches.

B.1. SSL Grounding

Similar to Section 4.4 in the main paper, we use Grad-CAM to evaluate the visual grounding ability of a contrastive SSL model trained with CAST and its effect on grounding in downstream tasks. Randomly sampled examples in Figure S2 show that the CAST-trained model seems to learn semantic category-specific feature representations, which allows it to look at objects of interest while performing query-key matching, and avoid learning spurious correlations.

B.2. Grounding of CASTed models on Downstream tasks

Fig. S3 shows randomly sampled qualitative examples showing how training with CAST also leads to improvements in grounding in downstream tasks. Notice in Fig. S3 (a, b) how the CASTed model looks at the whole extent of the object regions. We can also see in Fig. S3 (c) how in order to predict "Table Lamp", the MoCo pretrained model also looks at the table where as the CASTed model only looks at the relevant lamp regions. In Fig. S3 (f), in addition to not predicting the correct class "French Horn", the MoCo pretrained model is unable to localize the category, where as the CASTed model correctly predicts and attends to the "French Horn".

B.3. Qualitative examples from Background Challenge

The Backgrounds Challenge [1] (Sec 4.3 of the main text) aims to assess the background-robustness of image classification models by measuring their accuracy on images containing foreground objects superimposed on various background types (see [1] for details on dataset construction). Since CAST forces a model to attend to salient objects during learning, training with CAST also leads to models learning less spurious correlations, as seen in qualitative examples (Fig. S4). For example, in Fig. S4 (b), when the background corresponding to the sky is replaced by water, the MoCo pretrained model changes its decision from "Bird" to "Fish" indicating that the model was using spurious correlations associating water with fish. However the model pretrained with CAST, still correctly predicts the "Bird" class, indicating that it relies more on the foreground object of interest while making decisions.

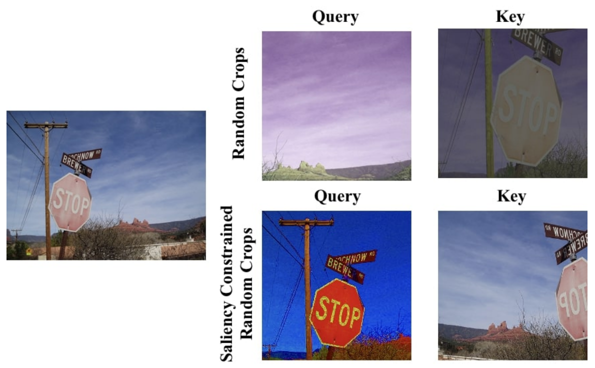
[*]Equal Contribution

Algorithm 1 Pseudocode of CAST applied on MoCo, PyTorch-style. CAST modifications highlighted in orange.

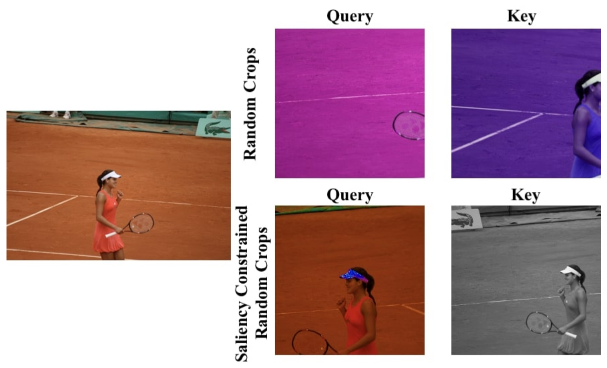
```
# f_query, f_key: Encoder networks for query and key
# queue: Dictionary as a queue of K keys (CxK)
# m: momentum
# t: temperature

# Initialize key network (momentum encoder) parameters as query network.
f_key.params = f_query.params

# Load a mini-batch of images and saliency maps of N instances.
for (x, s) in dataloader:
    # Generate saliency-constrained random crops (+ other data augmentations).
    x_q, s_q = aug(x, s) # For MoCo: x_q = aug(x)
    x_k, s_k = aug(x, s) # For MoCo: x_k = aug(x)

    # ------------------------------------------------------------------------------
    # Compute MoCo contrastive loss:
    # ------------------------------------------------------------------------------

    # Query and key feature vectors.
    q = f_query.forward(x_q) # . . . . . . . . . . . . . shape: (N, C)
    k = f_key.forward(x_k) # . . . . . . . . . . . . . . shape: (N, C)
    k = k.detach() # . . . . . . . . . . . . . . . . . . No gradient to keys.

    # Positive and negative logits.
    logits_pos = bmm(q.view(N, 1, C), k.view(N, C, 1)) # shape: (N, 1)
    logits_neg = bmm(q.view(N, C), queue.view(C, K)) # . shape: (N, K)

    logits = cat[logits_pos, logits_neg], dim=1) # . . . shape: (N, 1+K)

    # MoCo contrastive loss (Positive labels at index 0).
    labels = zeros(N)
    loss = CrossEntropyLoss(logits/t, labels)

    # ------------------------------------------------------------------------------
    # Compute CAST loss:
    # ------------------------------------------------------------------------------

    # Generate masked key and forward through momentum encoder.
    x_km = mask_op(x_k, s_k)
    km = f_key.forward(x_km) # . . . . . . . . . . . . . shape: (N, C)
    km = km.detach() # . . . . . . . . . . . . . . . . . No gradient to masked keys.

    # Compute dot product between query and masked key.
    dot_prod = mm(q.view(N, 1, C), km.view(N, C, 1)) # . shape: (N, 1)

    # Get activations from conv5 layer of query network.
    conv5_acts = lookup(f_query.params) #. . . . . . . . shape: (N, 7, 7, 2048)

    # Compute gradients of these conv5 activations wrt this dot product.
    conv5_grads = autograd.grad(dot_prod, conv5_acts) #. shape: (N, 7, 7, 2048)

    # Compute Grad-CAM map from these gradients.
    gradcam_map = compute_gradcam(conv5_grads) # . . . . shape: (N, 7, 7)

    # CAST Loss: conv5 Grad-CAM and query saliency map (maximize similarity).
    cast_loss = CosineSimilarityLoss(gradcam_map, resize7x7(s_q))
    loss += lambda * cast_loss
    loss.backward()

    # SGD update for query network.
    update(f_query.params)

    # Momentum update for key network.
    f_key.params = m * f_key.params + (1 - m) * f_query.params

    # Update dictionary (do not add masked keys).
    enqueue(queue, k) # Enqueue the current minibatch
    dequeue(queue) # Dequeue the earliest minibatch
```

References

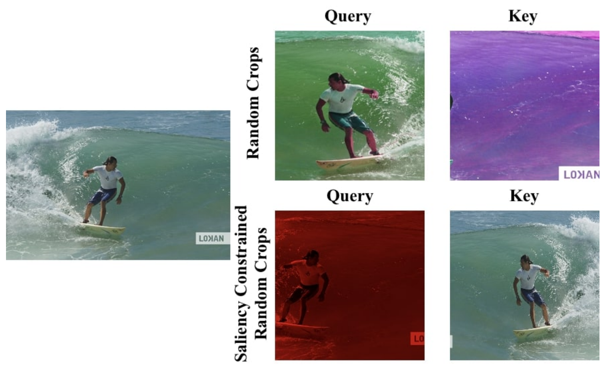
[1] Kai Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry, "Noise or signal: The role of image backgrounds in object recognition," *arXiv preprint arXiv:2006.09994*, 2020. 1



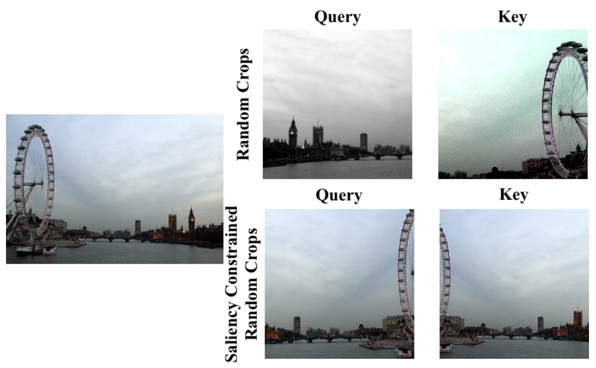
(a)



(b)



(c)



(d)

Figure S1: Saliency-constrained random crops always have salient regions that are overlapping between two randomly sampled crops. This can be compared to unconstrained random crops used in MoCo. Notice how in the random crops of (a), the stop sign (salient region) is missing from query, however query and key crops from our saliency-constrained random cropping approach contain a part of the stop sign. Similarly in (b) the salient region corresponding woman is missing from one of the crops, and in (c), the salient regions corresponding to the surfer is not present in both the query and key crops. The crops which do not contain overlapping salient regions (top-row in each subfigure) tend to provide noisy training signal to contrastive approaches forcing them to incorrectly produce similar features for crops containing varying context, e.g., stop sign and sky in (a), surfer and waves in (c), etc. Saliency-constrained random cropping mitigates this problem.

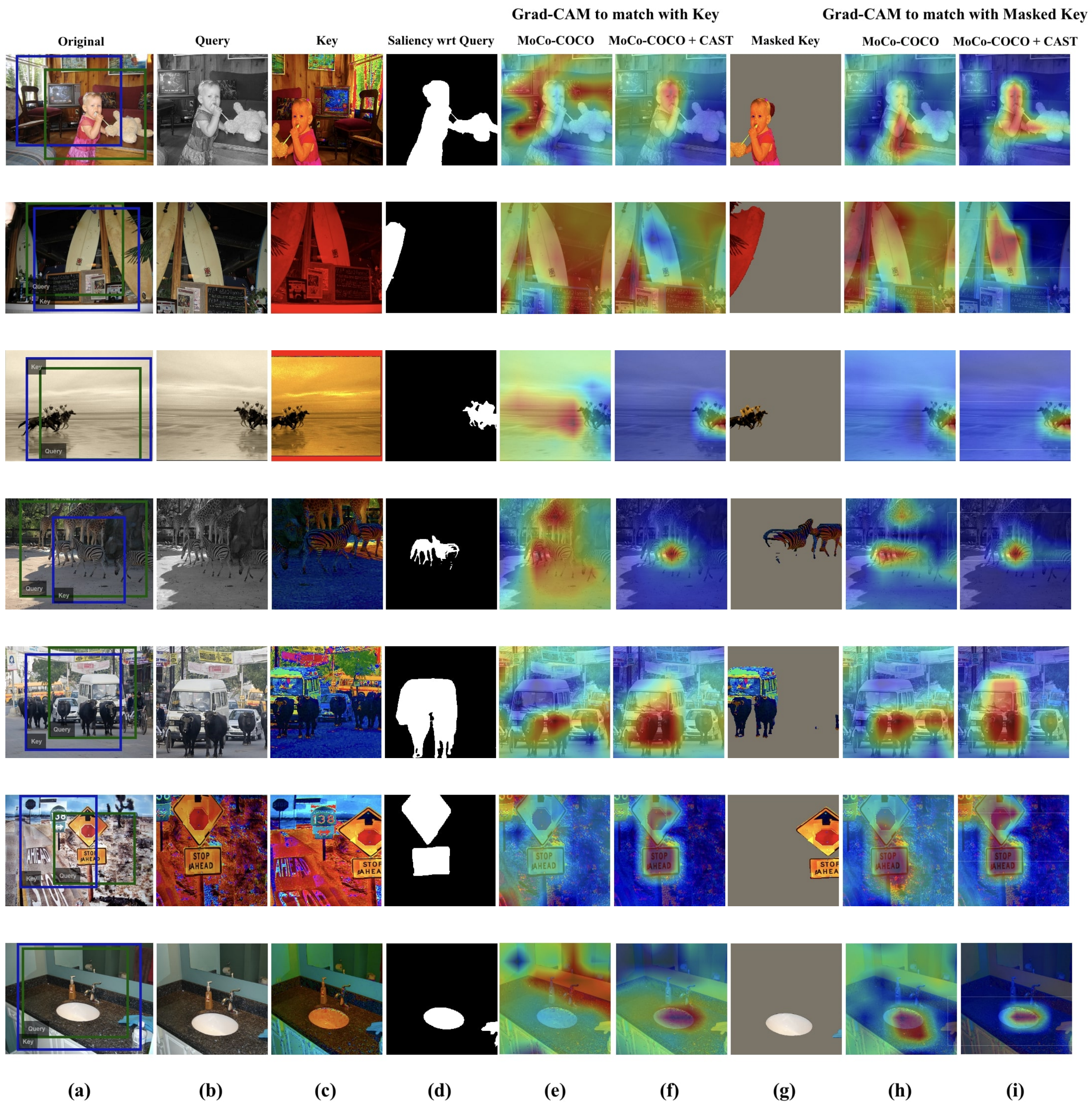


Figure S2: CAST improves visual grounding of the contrastive self-supervised feature encoder. Column (d) shows the saliency map according to the query crop (b). Grad-CAM visualizations in columns (e, f) show the query regions that MoCo and MoCo + CAST models rely on, in order to match the key crop (c). Finally, the MoCo and MoCo + CAST models rely on query regions (h) and (i) to match with the masked key representation (g). Notice how in all the examples, models trained with CAST look at the salient objects of interest to match the query with the key or the saliency-masked key.

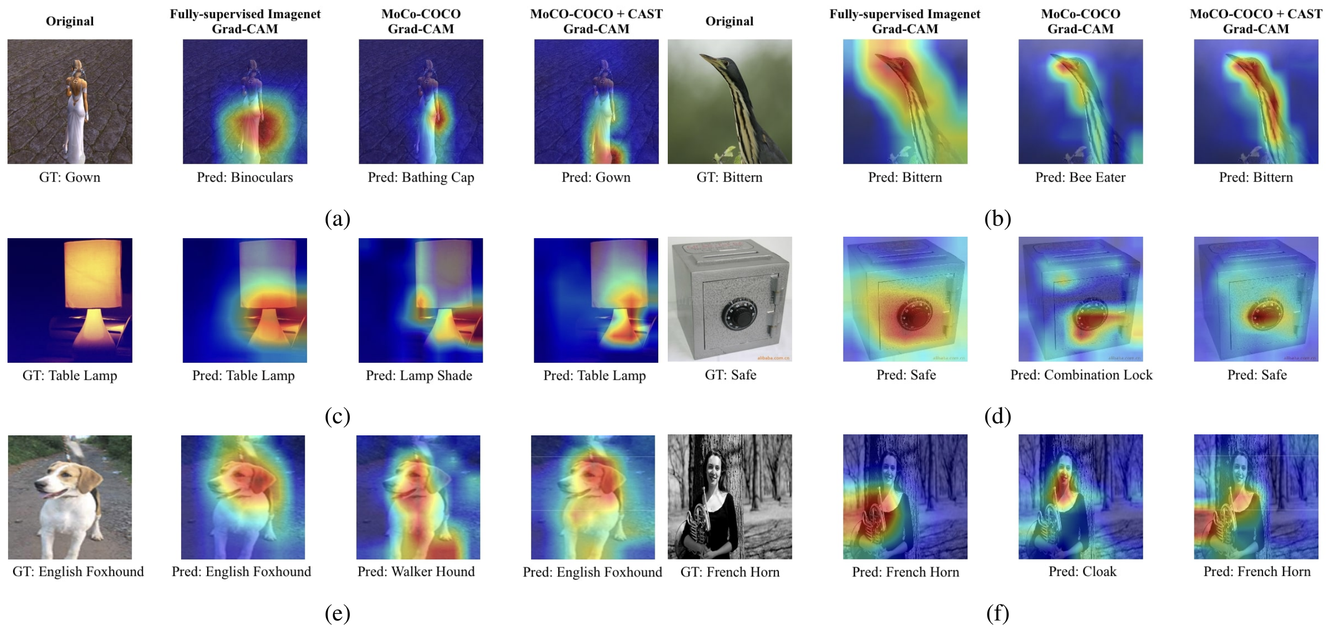
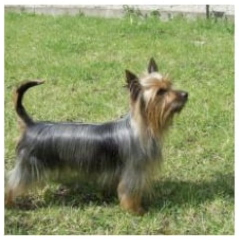


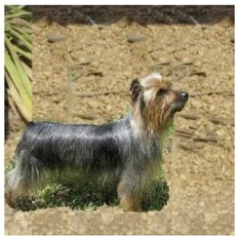
Figure S3: Downstream task Grounding. Qualitative comparison of Grad-CAM attention maps for the correct class from fully supervised networks and the self-supervised networks (MoCo and MoCo + CAST) on the Imagenet-1k classification task showing cases where the CAST fixes the mistakes made by MoCo model. We can see that CASTed models tend to look at the full extent of the object and less at the background regions. Note that even for incorrect predictions, Grad-CAM maps are computed for the ground-truth class.

Original



GT Class	Dog
Pred MoCo	Dog
Pred MoCo + CAST	Dog

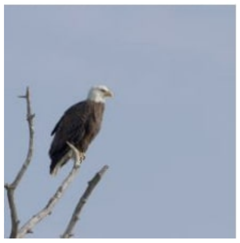
Mixed Rand



GT Class	Dog
Pred MoCo	Reptile
Pred MoCo + CAST	Dog

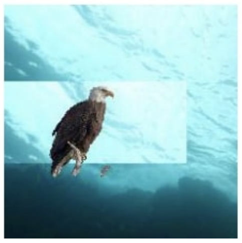
(a)

Original



GT Class	Bird
Pred MoCo	Bird
Pred MoCo + CAST	Bird

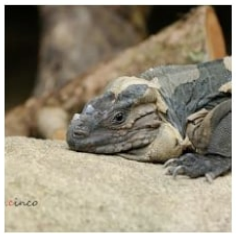
Mixed Rand



GT Class	Bird
Pred MoCo	Fish
Pred MoCo + CAST	Bird

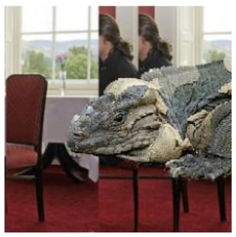
(b)

Original



GT Class	Reptile
Pred MoCo	Reptile
Pred MoCo + CAST	Reptile

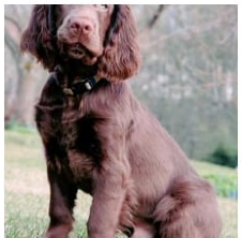
Mixed Rand



GT Class	Reptile
Pred MoCo	Instrument
Pred MoCo + CAST	Reptile

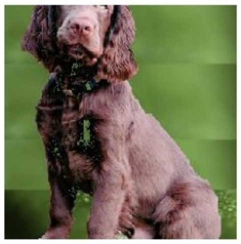
(c)

Original



GT Class	Dog
Pred MoCo	Dog
Pred MoCo + CAST	Dog

Mixed Rand



GT Class	Dog
Pred MoCo	Bird
Pred MoCo + CAST	Dog

(d)

Figure S4: Backgrounds-challenge. Models pretrained with CAST are less likely to rely on background correlations for classification. Notice how in (a), the MoCo pretrained model incorrectly predicts "Reptile" when the grass is changed to a background from a reptile class, while the CAST pretrained model still correctly predicts "Dog". Similarly in (c), when chairs are introduced in the background, the MoCo pretrained model changes its prediction from "Reptile" to "Instrument", indicating that the model relies on spurious correlations to predict the class of interest. The CAST pretrained model still correctly predicts "Reptile."