Hydra Attention: Efficient Attention with Many Heads Appendix

Daniel Bolya[1,2⋆], Cheng-Yang Fu[2], Xiaoliang Dai[2], Peizhao Zhang[2], and Judy Hoffman[1]

[1] Georgia Tech
{dbolya,judy}@gatech.edu
[2] Meta AI
{chengyangfu,xiaoliangdai,stzpz}@fb.com

1 Other Kernels

In Tab. 1, we list all the kernel functions (instantiations of $\phi(\cdot)$) we've tried with Hydra Attention. There are three main concerns we had while choosing these kernels. Namely, should $\phi(\cdot)$ be 1.) unbounded, 2.) allow for negative values, or 3.) be linear.

L2 normalization for cosine similarity (which is what we use in the paper), for instance, is bounded, allows for negatives, and is linear. Sigmoid, on the other hand, is bounded, is only positive, and is non-linear.

From our experiments, we've observed that while the function used for Q is not that important, K significantly benefits from being both linear and allowing negative values. Compared to L2 normalization, softmaxing K significantly degrades performance. And out of the normalization techniques, we find L2 to work the best (over L1 or constant normalization).

⋆ This work was done under an internship at Meta AI.

Kernel	$\phi(Q)$	$\phi(K)$	Accuracy
Cosine Similarity	$x/\|\|x\|\|_2$		**76.37**
Tanh-L2	$\tanh(x)$	$x/\|\|x\|\|_2$	76.17
Mean	$x/\sqrt{T}$		75.95
CosSim + LN	$x/\|\|x\|\|_2$		75.74
Tanh-L2 + LN	$\tanh(x)$	$x/\|\|x\|\|_2$	75.22
Tanh-Softmax	$\tanh(x)$	$\text{softmax}(x)$	74.18
Sigmoid-Softmax	$\sigma(x)$	$\text{softmax}(x)$	74.02
L1 Normalization	$x/\|\|x\|\|_1$		70.75

Table 1: **More Kernels.** We include other kernels we've tested here. Since most are asymmetric, we list ϕ for Q and K separately.

Finally, since $\phi(K)$ (when not softmax) does not sum to 1, multiplying it by V can produce magnitudes higher than standard attention. Thus, we thought it might be useful to normalize the result of the attention layer. We test two kernels with "+ LN", where the attention operation is followed by a Layer Norm (before the projection). This, however, does not seem to help, so it seems better to leave the kernel unnormalized here.

2 More Visualizations

In Fig. 1, we include several more image visualizations to supplement those in the main paper. These images were selected randomly from 12 different classes (4 per class) from the ImageNet-1k validation set with the only selection criteria being that the image is safe to view. The network predicts most of these images correctly.

3 Code

Hydra attention is extremely simple to implement. We give a reference implementation with the cosine similarity kernel here in PyTorch:

```
def hydra(q, k, v):
    """
    q, k, and v should all be tensors of shape
        [batch, tokens, features]
    """
    q = q / q.norm(dim=-1, keepdim=True)
    k = k / k.norm(dim=-1, keepdim=True)

    kv = (k * v).sum(dim=-2, keepdim=True)
    out = q * kv

    return out
```

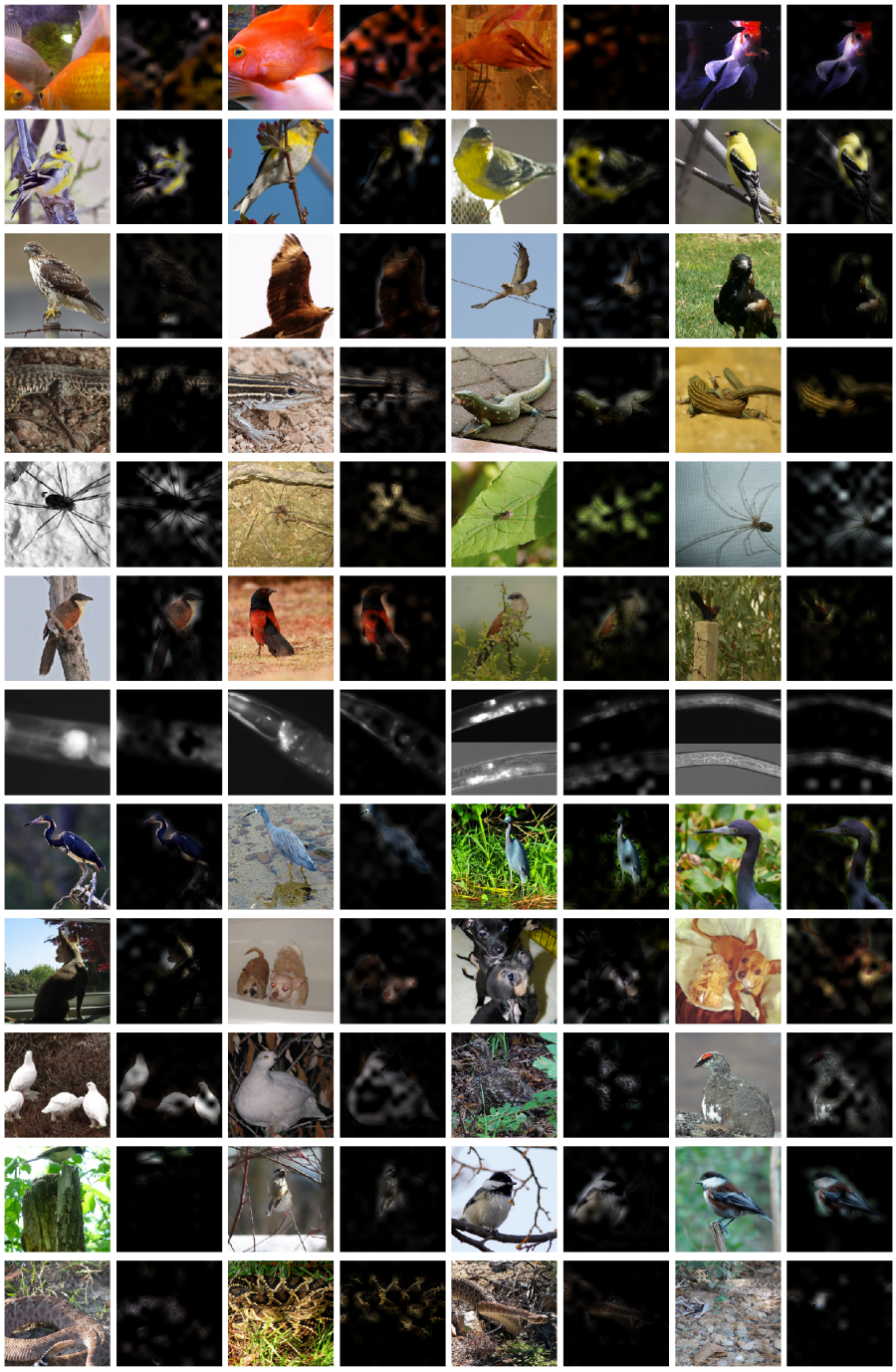

Fig. 1: **More Visualization.** More visualization of the focus of the model using the method described in the paper.