Keep It SimPool:
Who Said Supervised Transformers Suffer from Attention Deficit?
Supplementary Material

Bill Psomas[1,2] Ioannis Kakogeorgiou[1] Konstantinos Karantzalos[1] Yannis Avrithis[2]

[1]National Technical University of Athens
[2]Institute of Advanced Research in Artificial Intelligence (IARAI)

A1. Extended related work

Spatial pooling of visual input is the process by which spatial resolution is reduced to 1×1, such that the input is mapped to a single vector. This process can be gradual and interleaved with mapping to a feature space, because any feature space is amenable to smoothing or downsampling. The objective is robustness to deformation while preserving important visual information.

Via a similarity function, *e.g.* dot product, the vector representation of an image can be used for efficient matching to class representations for category-level tasks or to the representation of another image for instance-level tasks. One may obtain more than one vectors per image as a representation, but this requires a particular kernel for matching.

Background The study of receptive fields in neuroscience [15] lead to the development of 2D *Gabor filters* [16] as a model of the first processing layer in the visual cortex. Visual descriptors based on filter banks in the frequency domain [61] and orientation histograms [51, 14] can be seen as efficient implementations of the same idea. Apart from mapping to a new space—that of filter responses or orientation bins—they involve a form of smoothing, at least in some orientation, and weighted local spatial pooling.

Textons [17] can be seen as a second layer, originally studied in the context of texture discrimination [81] and segmentation [17, 52] and taking the form of multidimensional histograms on Gabor filter responses. The *bag of words* model [75, 12] is based on the same idea, as a histogram on other visual descriptors. Again, apart from mapping to a new space—that of textons or visual words—they involve local or global spatial pooling.

Histograms and every step of building visual features can be seen as a form of nonlinear coding followed by pooling [5]. *Coding* is maybe the most important factor. For example, a high-dimensional mapping before pooling, optionally followed by dimension reduction after pooling, can reduce interference between elements [62, 35, 4]. *Weighting* of individual elements is also important in attending important regions [25, 34, 76] and in preventing certain elements from dominating others [32, 54, 33].

The *pooling operation* itself is any symmetric (permutation-invariant) set function, which can be expressed in the form $F(X) = g\left(\sum_{x \in X} f(x)\right)$ [98]. The most common is average and maximum [72, 6, 5].

Common ways to obtain a representation of *multiple vectors* are using a spatial partition [25] or a partition in the feature space [31, 77].

Convolutional networks Following findings of neuroscience, early convolutional networks [22, 42] are based on learnable *convolutional layers* interleaved with fixed *spatial pooling layers* that downsample, which is an instance of the coding-pooling framework. The same design remains until today [41, 74, 26, 47]. Again, apart from mapping to a new space, convolutional layers involve a form of weighted local pooling. Again, the operation in pooling layers is commonly average [42] or maximum [72, 41].

Early networks end in a fully-connected layer over a feature tensor of low resolution [42, 41, 74]. This evolved into spatial pooling, *e.g. global average pooling* (GAP) for classification [43, 26], regional pooling for detection [23], or global maximum followed by a pairwise loss [79] for instance-level tasks. This is beneficial for downstream tasks and interpretability [101].

The spatial pooling operation at the end of the network is widely studied in instance level-tasks [2, 79, 69], giving rise to forms of *spatial attention* [37, 59, 7, 78, 57], In category-level tasks, it is more common to study *feature re-weighting* as components of the architecture [29, 92, 28]. The two are closely related because *e.g.* the weighted average is element-wise weighting followed by sum. Most modern pooling operations are learnable.

Pooling can be *spatial* [28, 59, 7, 78, 57], *over channels* [29], or both [37, 92]. CBAM [92] is particularly related to our work in the sense that it includes global average pooling followed by a form of spatial attention, although the latter is not evident in its original formulation and although CBAM is designed as a feature re-weighting rather than pooling mechanism.

One may obtain a representation of *multiple vectors e.g.* by some form of clustering [30] or optimal transport [53].

Vision transformers Pairwise interactions between features are forms of *self-attention* that can be seen as alternatives to convolution or forms of pooling. They have commonly been designed as architectural components of convolutional networks, again over the spatial [90, 3, 100, 67] or the channel dimensions [9, 87]. Originating in language models [83], *vision transformers* [18] streamlined these approaches and became the dominant competitors of convolutional networks.

Transformers commonly downsample only at the input, forming spatial *patch tokens*. Pooling is based on a learnable CLS ("classification") token, which, beginning at the input space, undergoes the same self-attention operation with patch tokens and eventually provides a global image representation. That is, the network ends in global weighted average pooling, using as weights the attention of CLS over the patch tokens. Pooling is still gradual, since CLS interacts with patch tokens throughout the network depth.

Several variants of transformers often bring back ideas from convolutional networks, including spatial hierarchy [45], relative position encoding [94, 24], re-introducing convolution [93, 19], re-introducing pooling layers [27, 45, 88, 89], or simple pooling instead of attention [97]. In this sense, downsampling may occur inside the transformer, *e.g.* for classification [27, 45] or detection [88, 89].

Few works that have studied anything other than CLS for pooling in transformers are mostly limited to GAP [45, 99, 82, 70]. CLS offers attention maps for free, but those are typically of low quality unless in a self-supervised setting [8], which is not well studied. Few works that attempt to rectify this in the supervised setting include a spatial entropy loss [63], shape distillation from convolutional networks [56] and skipping computation of self-attention, observing that the quality of self-attention is still good at intermediate layers [84]. It has also been found beneficial to inject the CLS token only at the last few layers [80].

We are thus motivated to question why the pooling operation at the end of the network needs to be different in convolutional networks and vision transformers and why pooling with a CLS token needs to be performed across the network depth. We study pooling in both kinds of networks, in supervised and self-supervised settings alike. We derive a simple, attention-based, universal pooling mechanism that applies equally to all cases, improving both performance and the quality of attention maps.

A2. More on the method

In subsection A2.1, we summarize the generalized pooling framework of subsection 3.1. We then detail how to formulate methods studied in subsection 3.2 as instances of our pooling framework so as to obtain Table 1, examining them in groups as in subsection 3.2. Finally, we summarize SimPool in subsection A2.6.

Notation By id we denote the identity mapping. Given $n \in \mathbb{N}$, we define $[n] := \{1, \dots, n\}$. By $\mathbb{1}_A$ we denote the indicator function of set A, by δ_{ij} the Kronecker delta and by $[P]$ the Iverson bracket of statement P. By $A \circ B$ we denote the Hadamard product of matrices A, B and by $A^{\circ n}$ the Hadamard n-th power of A. We recall that by η_1, η_2 we denote the row-wise and column-wise ℓ_1-normalization of a matrix, respectively, while $\boldsymbol{\sigma}_2$ is column-wise softmax.

Algorithm 1: Our generalized pooling framework.

```
input  : $p$: #patches, $d$: dimension
input  : $X \in \mathbb{R}^{d \times p}$: features
option : $k$: #pooled vectors
option : INIT: pooling initialization
option : $T$: #iterations
option : $\{\phi_Q^t\}, \{\phi_K^t\}$: query, key mappings
option : $s$: pairwise similarity function
option : $h$: attention function
option : $\{\phi_V^t\}$: value mapping
option : $f$: pooling function
option : $\{\phi_X^t\}, \{\phi_U^t\}$: output mappings
output : $d'$: output dimension
output : $U \in \mathbb{R}^{d' \times k}$: pooled vectors

$d^0 \leftarrow d$                                          ▷ input dimension
$X^0 \leftarrow X \in \mathbb{R}^{d^0 \times k}$            ▷ initialize features
$U^0 \leftarrow \textsc{init}(X) \in \mathbb{R}^{d^0 \times k}$   ▷ initialize pooling
for $t = 0, \dots, T-1$ do
    $Q \leftarrow \phi_Q^t(U^t) \in \mathbb{R}^{n^t \times k}$              ▷ query (1)
    $K \leftarrow \phi_K^t(X^t) \in \mathbb{R}^{n^t \times p}$              ▷ key (2)
    $S \leftarrow \mathbf{0}_{p \times k}$                                  ▷ pairwise similarity
    for $i \in [p]$, $j \in [k]$ do
        $s_{ij} \leftarrow s(\mathbf{k}_{\bullet i}, \mathbf{q}_{\bullet j})$
    $A \leftarrow h(S) \in \mathbb{R}^{p \times k}$                         ▷ attention (3)
    $V \leftarrow \phi_V^t(X^t) \in \mathbb{R}^{n^t \times p}$              ▷ value (4)
    $Z \leftarrow f^{-1}(f(V)A) \in \mathbb{R}^{n^t \times k}$              ▷ pooling (5)
    $X^{t+1} \leftarrow \phi_X^t(X^t) \in \mathbb{R}^{d^{t+1} \times p}$    ▷ update feat. (6)
    $U^{t+1} \leftarrow \phi_U^t(Z) \in \mathbb{R}^{d^{t+1} \times k}$      ▷ update pool. (7)
$d' \leftarrow d^T$                                         ▷ output dimension
$U \leftarrow U^T$                                          ▷ pooled vectors
```

A2.1. Pooling framework summary

Our generalized pooling framework is summarized in algorithm 1. As *input*, it takes the features $X \in \mathbb{R}^{d \times p}$, representing p patch embeddings of dimension d. As *output*, it returns the pooled vectors $U \in \mathbb{R}^{d' \times k}$, that is, k vectors of dimension d'. As *options*, it takes the number k of vectors to pool; the pooling initialization function INIT; the number T of iterations; the query and key mappings $\{\phi_Q^t\}, \{\phi_K^t\}$; the pairwise similarity function s; the attention function h; the value mapping $\{\phi_V^t\}$; the pooling function f; and the output mappings $\{\phi_X^t\}, \{\phi_U^t\}$.

The mappings and dimensions within iterations may be different at each iteration, and all optional functions may be learnable. As such, the algorithm is general enough to incorporate any deep neural network. However, the focus is on pooling, as is evident by the pairwise similarity between queries (pooled vectors) and keys (features) in line 9, which is a form of *cross-attention*.

A2.2. Group 1: Simple methods with $k = 1$

These methods are non-iterative, there are no query Q, key K, similarity matrix S or function h, and the attention is a vector $\mathbf{a} \in \mathbb{R}^p$ that is either fixed or a function directly of X. With the exception of HOW [78], the value matrix is $V = X$, that is, $\phi_V = \mathrm{id}$, and we are pooling into vector $\mathbf{u} = \mathbf{z} \in \mathbb{R}^d$, that is, $\phi_U = \mathrm{id}$. Then, (5) takes the form

$$\mathbf{u} = f^{-1}(f(X)\mathbf{a}) \in \mathbb{R}^d, \tag{A1}$$

and we focus on instantiating it to identify function f and attention vector $\mathbf{a}$. With the exception of LSE [66], function f is f_α (8) and we seek to identify α.

Global average pooling (GAP) [43, 26] According to (9),

$$\pi_A(X) := \frac{1}{p}\sum_{j=1}^{p} \mathbf{x}_{\bullet j} = X\mathbf{1}_p/p = f_{-1}^{-1}(f_{-1}(X)\mathbf{a}), \tag{A2}$$

where $f_{-1}(x) = x^{\frac{1-(-1)}{2}} = x$, thus $f_{-1} = \mathrm{id}$, and $\mathbf{a} = \mathbf{1}_p/p$.

Max pooling [79] Assuming $X \geq 0$,

$$\begin{aligned}\pi_{\max}(X) &:= \max_{j \in [p]} \mathbf{x}_{\bullet j} = \lim_{\gamma \to \infty} \left(\sum_{j=1}^{p} \mathbf{x}_{\bullet j}^{\gamma} \right)^{\frac{1}{\gamma}} && \text{(A3)}\\ &= \lim_{\gamma \to \infty} (X^\gamma \mathbf{1}_p)^{\frac{1}{\gamma}} = f_{-\infty}^{-1}(f_{-\infty}(X)\mathbf{a}), && \text{(A4)}\end{aligned}$$

where all operations are taken element-wise and $\mathbf{a} = \mathbf{1}_p$.

Generalized mean (GeM) [69] Assuming $X \geq 0$,

$$\begin{aligned}\pi_{\mathrm{GeM}}(X) &:= \left(\frac{1}{p}\sum_{j=1}^{p} \mathbf{x}_{\bullet j}^{\gamma} \right)^{\frac{1}{\gamma}} && \text{(A5)}\\ &= (X^\gamma \mathbf{1}_p/p)^{\frac{1}{\gamma}} = f_\alpha^{-1}(f_\alpha(X)\mathbf{a}), && \text{(A6)}\end{aligned}$$

where all operations are taken element-wise, $\gamma = (1-\alpha)/2$ is a learnable parameter and $\mathbf{a} = \mathbf{1}_p/p$.

SimPool has the same pooling function but is based on an attention mechanism.

Log-sum-exp (LSE) [66]

$$\begin{aligned}\pi_{\mathrm{LSE}}(X) &:= \frac{1}{r}\log\left(\frac{1}{p}\sum_{j=1}^{p} \exp(r\mathbf{x}_{\bullet j}) \right) && \text{(A7)}\\ &= f^{-1}(f(X)\mathbf{a}), && \text{(A8)}\end{aligned}$$

where all operations are taken element-wise, r is a learnable scale parameter, $f(x) = e^{rx}$ and $\mathbf{a} = \mathbf{1}_p/p$.

HOW [78] The attention value of each feature $\mathbf{x}_{\bullet j}$ is its norm $\|\mathbf{x}_{\bullet j}\|$. That is,

$$\begin{aligned}\mathbf{a} &= (\|\mathbf{x}_{\bullet 1}\|, \dots, \|\mathbf{x}_{\bullet p}\|)^\top = (X^{\circ 2})^\top \mathbf{1}_d && \text{(A9)}\\ &= \mathrm{diag}(X^\top X) \in \mathbb{R}^p, && \text{(A10)}\end{aligned}$$

obtained by pooling over channels. The value matrix is

$$V = \phi_V(X) = \textsc{fc}(\mathrm{avg}_3(X)) \in \mathbb{R}^{d' \times p}, \tag{A11}$$

where avg_3 is 3×3 local average pooling, FC is a fixed fully-connected (1×1 convolutional) layer incorporating centering, PCA dimension reduction and whitening according to the statistics of the local features of the training set and $d' < d$ is the output dimension. Then,

$$\mathbf{z} = \sum_{j=1}^{p} a_j \mathbf{v}_{\bullet j} = V\mathbf{a} = f_{-1}^{-1}(f_{-1}(V)\mathbf{a}) \in \mathbb{R}^{d'}, \tag{A12}$$

where $f_{-1} = \mathrm{id}$ as in GAP. Finally, the output is $\mathbf{u} = \eta^2(\mathbf{z})$, where the mapping $\phi_U = \eta^2$ is ℓ_2-normalization.

A2.3. Group 2: Iterative methods with $k > 1$

We examine three methods, which, given $X \in \mathbb{R}^{d \times p}$ and $k < p$, seek $U \in \mathbb{R}^{d \times k}$ by iteratively optimizing a kind of assignment between columns of X and U. The latter are called references [53], centroids [48], or slots [49]. Assignment can be soft [53, 49] or hard [48]. It can be an assignment of columns of X to columns of U [48, 49] or both ways [53]. The algorithm may contain learnable components [53, 49] or not [48].

Optimal transport kernel embedding (OTK) [53] Pooling is based on a learnable parameter $U \in \mathbb{R}^{d \times k}$. We define the $p \times k$ *cost* matrix $C = (c_{ij})$ consisting of the pairwise squared Euclidean distances between columns of X and U, *i.e.*, $c_{ij} = \|\mathbf{x}_{\bullet i} - \mathbf{u}_{\bullet j}\|^2$. We seek a $p \times k$ non-negative *transportation plan* matrix $P \in \mathcal{P}$ representing a joint probability distribution over features of X and U with uniform marginals:

$$\mathcal{P} := \{P \in \mathbb{R}_+^{p \times k} : P\mathbf{1}_k = \mathbf{1}_p/p, P^\top \mathbf{1}_p = \mathbf{1}_k/k\}. \tag{A13}$$

The objective is to minimize the expected, under P, pairwise cost with entropic regularization

$$P^* := \arg\min_{P \in \mathcal{P}} \langle P, C \rangle - \epsilon H(P), \tag{A14}$$

where $H(P) = -\mathbf{1}_p^\top (P \circ \log P)\mathbf{1}_k$ is the entropy of P, $\langle \cdot, \cdot \rangle$ is the Frobenius inner product and $\epsilon > 0$ controls the sparsity of P. The optimal solution is $P^* = \textsc{Sinkhorn}(e^{-C/\epsilon})$, where exponentiation is element-wise and $\textsc{Sinkhorn}$ is the Sinkhorn-Knopp algorithm [38], which iteratively ℓ_1-normalizes rows and columns of a matrix until convergence [13]. Finally, pooling is defined as

$$U = \psi(X)P^* \in \mathbb{R}^{d' \times k}, \tag{A15}$$

where $\psi(X) \in \mathbb{R}^{d' \times p}$ and $\psi : \mathbb{R}^d \to \mathbb{R}^{d'}$ is a Nyström approximation of a kernel embedding in $\mathbb{R}^d$, *e.g.* a Gaussian kernel [53], which applies column-wise to $X \in \mathbb{R}^{d \times p}$.

We conclude that OTK [53] is a instance of our pooling framework with learnable $U_0 = U \in \mathbb{R}^{d \times k}$, query/key mappings $\phi_Q = \phi_K = \text{id}$, pairwise similarity function $s(\mathbf{x}, \mathbf{y}) = -\|\mathbf{x} - \mathbf{y}\|^2$, attention matrix $A = h(S) = \textsc{Sinkhorn}(e^{S/\epsilon}) \in \mathbb{R}^{p \times k}$, value mapping $\phi_V = \psi$, average pooling function $f = f_{-1}$ and output mapping $\phi_U = \text{id}$.

Although OTK is not formally iterative in our framework, $\textsc{Sinkhorn}$ internally iterates indeed to find a soft-assignment between the features of X and U.

***k*-means [48]** k-means aims to find a $d \times k$ matrix U minimizing the sum of squared Euclidean distances of each column $\mathbf{x}_{\bullet i}$ of X to its nearest column $\mathbf{u}_{\bullet j}$ of U:

$$J(U) := \sum_{i=1}^{p} \min_{j \in [k]} \|\mathbf{x}_{\bullet i} - \mathbf{u}_{\bullet j}\|^2. \tag{A16}$$

Observe that (10) is the special case $k = 1$, where the unique minimum $\mathbf{u}^* = \pi_A(X)$ is found in closed form (11). For $k > 1$, the distortion measure J is non-convex and we are only looking for a local minimum.

The standard k-means algorithm is initialized by a $d \times k$ matrix U^0 whose columns are k of the columns of X sampled at random and represent a set of k *centroids* in $\mathbb{R}^d$. Given U^t at iteration t, we define the $p \times k$ *distance* matrix $D = (d_{ij})$ consisting of the pairwise squared Euclidean distances between columns of X and U^t, *i.e.*, $d_{ij} = \left\|\mathbf{x}_{\bullet i} - \mathbf{u}_{\bullet j}^t\right\|^2$. For $i \in [p]$, feature $\mathbf{x}_{\bullet i}$ is *assigned* to the nearest centroid $\mathbf{u}_{\bullet j}^t$ with index

$$c_i = \arg\min_{j \in [k]} d_{ij}, \tag{A17}$$

where ties are resolved to the lowest index. Then, at iteration $t + 1$, centroid $\mathbf{u}_{\bullet j}^t$ is *updated* as the mean of features $\mathbf{x}_{\bullet i}$ assigned to it, *i.e.*, for which $c_i = j$:

$$\mathbf{u}_{\bullet j}^{t+1} = \frac{1}{\sum_{i=1}^p \delta_{c_i j}} \sum_{i=1}^{p} \delta_{c_i j} \mathbf{x}_{\bullet i}. \tag{A18}$$

Let $\arg\min_1(D)$ be the $p \times k$ matrix $M = (m_{ij})$ with

$$m_{ij} = \delta_{c_i j} = \left[j = \arg\min_{j' \in [k]} d_{ij'}\right]. \tag{A19}$$

That is, each row $\mathbf{d}_i \in \mathbb{R}^k$ of D yields a row $\mathbf{m}_i \in \{0, 1\}^k$ of M that is an one-hot vector indicating the minimal element over $\mathbf{d}_i$. Define operator $\arg\max_1$ accordingly. Then, (A18) can be written in matrix form as

$$U^{t+1} = X\eta_2(\arg\max_1(-D)) \in \mathbb{R}^{d \times k}. \tag{A20}$$

We conclude that k-means is an iterative instance of our pooling framework with the columns of $U^0 \in \mathbb{R}^{d \times k}$ sampled at random from the columns of X, query/key mappings $\phi_Q = \phi_K = \text{id}$, pairwise similarity function $s(\mathbf{x}, \mathbf{y}) = -\|\mathbf{x} - \mathbf{y}\|^2$, attention matrix $A = h(S) = \eta_2(\arg\max_1(S)) \in \mathbb{R}^{p \times k}$, value mapping $\phi_V = \text{id}$, average pooling function $f = f_{-1}$ and output mappings $\phi_X = \phi_U = \text{id}$.

Slot attention [49] Pooling is initialized by a random $d' \times k$ matrix U^0 sampled from a normal distribution $\mathcal{N}(\mu, \sigma^2)$ with shared, learnable mean $\mu \in \mathbb{R}^{d'}$ and standard deviation $\sigma \in \mathbb{R}^{d'}$. Given U^t at iteration t, define the query $Q = W_Q \textsc{ln}(U^t) \in \mathbb{R}^{n \times k}$ and key $K = W_K \textsc{ln}(X) \in \mathbb{R}^{n \times p}$, where $\textsc{ln}$ is LayerNorm [1] and n is a common dimension. An attention matrix is defined as

$$A = \eta_1(\boldsymbol{\sigma}_2(K^\top Q/\sqrt{n})) \in \mathbb{R}^{p \times k}. \tag{A21}$$

Then, with value $V = W_V \textsc{ln}(X) \in \mathbb{R}^{n \times p}$, pooling is defined as the weighted average

$$Z = VA \in \mathbb{R}^{n \times k}. \tag{A22}$$

Finally, U^t is updated according to

$$G = \text{GRU}(Z) \in \mathbb{R}^{d' \times k} \tag{A23}$$

$$U^{t+1} = G + \text{MLP}(\text{LN}(G)) \in \mathbb{R}^{d' \times k}, \tag{A24}$$

where GRU is a *gated recurrent unit* [10] and MLP a multi-layer perceptron with ReLU activation and a residual connection [49].

We now simplify the above formulation by removing LayerNorm and residual connections.

> We conclude that slot attention [49] is an iterative instance of our pooling framework with U^0 a random $d' \times k$ matrix sampled from $\mathcal{N}(\mu, \sigma^2)$ with learnable parameters $\mu, \sigma \in \mathbb{R}^{d'}$, query mapping $\phi_Q(U) = W_Q U \in \mathbb{R}^{n \times k}$, key mapping $\phi_K(X) = W_K X \in \mathbb{R}^{n \times p}$, pairwise similarity function $s(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}$, attention matrix $A = h(S) = \eta_1(\boldsymbol{\sigma}_2(S/\sqrt{n})) \in \mathbb{R}^{p \times k}$, value mapping $\phi_V(X) = W_V X \in \mathbb{R}^{n \times p}$, average pooling function $f = f_{-1}$, output mapping $\phi_U(Z) = \text{MLP}(\text{GRU}(Z)) \in \mathbb{R}^{d' \times k}$ and output dimension d'.

SimPool is similar in its attention mechanism, but is non-iterative with $k = 1$ and initialized by GAP.

A2.4. Group 3: Feature re-weighting, $k = 1$

We examine two methods, originally proposed as components of the architecture, which use attention mechanisms to re-weight features in the channel or the spatial dimension. We modify them by placing at the end of the network, followed by GAP. We thus reveal that they serve as attention-based pooling. This includes pairwise interaction, although this was not evident in their original formulation.

Squeeze-and-excitation block (SE) [29] The *squeeze* operation aims to mitigate the limited receptive field of convolutional networks, especially in the lower layers. It uses global average pooling over the spatial dimension,

$$\mathbf{u}^0 = \pi_A(X) \in \mathbb{R}^d. \tag{A25}$$

Then, the *excitation* operation aims at capturing channel-wise dependencies and involves two steps. In the first step, a learnable gating mechanism forms a vector

$$\mathbf{q} = \sigma(\text{MLP}(\mathbf{u}^0)) \in \mathbb{R}^d, \tag{A26}$$

where σ is the sigmoid function and MLP concists of two linear layers with ReLU activation in-between and forming a bottlenect of hidden dimension d/r. This vector expresses an importance of each channel that is not mutually exclusive. The second step re-scales each channel (row) of X by the corresponding element of $\mathbf{q}$,

$$V = \text{diag}(\mathbf{q}) X \in \mathbb{R}^{d \times p}. \tag{A27}$$

The output $X' = V \in \mathbb{R}^{d \times p}$ is a new tensor of the same shape as X, which can be used in the next layer. In this sense, the entire process is considered a block to be used within the architecture of convolutional networks at several layers. This yields a new family of networks, called *squeeze-and-excitation networks* (SENet).

However, we can also see it as a pooling process if we perform it at the end of a network, followed by GAP:

$$\mathbf{z} = \pi_A(V) = \text{diag}(\mathbf{q}) X \mathbf{1}_p / p \in \mathbb{R}^d, \tag{A28}$$

> We conclude that this modified SE block is a non-iterative instance of our pooling framework with $\mathbf{u}^0 = \pi_A(X) \in \mathbb{R}^d$, query mapping $\phi_Q(\mathbf{u}) = \sigma(\text{MLP}(\mathbf{u})) \in \mathbb{R}^d$, no key K, similarity matrix S of function h, uniform spatial attention $\mathbf{a} = \mathbf{1}_p/p$, value mapping $\phi_V(X) = \text{diag}(\mathbf{q}) X \in \mathbb{R}^{d \times p}$ and average pooling function $f = f_{-1}$.

The original design does not use $\mathbf{a}$ or $\mathbf{z}$; instead, it has an output mapping $\phi_X(X) = V = \text{diag}(\mathbf{q}) X \in \mathbb{R}^{d \times p}$. Thus, it can be used iteratively along with other mappings of X to form a modified network architecture.

Convolutional block attention module (CBAM) [92] This is an extension of SE [29] that acts on both the channel and spatial dimension in similar ways. *Channel attention* is similar to SE: It involves (a) global average and maximum pooling of X over the spatial dimension,

$$U^0 = (\pi_A(X) \;\; \pi_{\max}(X)) \in \mathbb{R}^{d \times 2}; \tag{A29}$$

(b) a learnable gating mechanism forming vector

$$\mathbf{q} = \sigma(\text{MLP}(U^0) \mathbf{1}_2 / 2) \in \mathbb{R}^d, \tag{A30}$$

which is defined as in SE [29] but includes averaging over the two columns before σ; and (c) re-scaling channels (rows) of X by $\mathbf{q}$,

$$V = \text{diag}(\mathbf{q}) X \in \mathbb{R}^{d \times p}. \tag{A31}$$

Spatial attention performs a similar operation in the spatial dimension: (a) global average and maximum pooling of V over the channel dimension,

$$S = (\pi_A(V^\top) \;\; \pi_{\max}(V^\top)) \in \mathbb{R}^{p \times 2}; \tag{A32}$$

(b) a learnable gating mechanism forming vector

$$\mathbf{a} = \sigma(\text{conv}_7(S)) \in \mathbb{R}^p, \tag{A33}$$

where conv_7 is a a convolutional layer with kernel size 7×7; and (c) re-scaling features (columns) of V by $\mathbf{a}$,

$$X' = V \, \text{diag}(\mathbf{a}) \in \mathbb{R}^{d \times p}. \tag{A34}$$

The output X' is a new tensor of the same shape as X, which can be used in the next layer. In this sense, CBAM is a block to be used within the architecture, like SE [29]. However, we can also see it as a *pooling process* if we perform it at the end of a network, followed by GAP:

$$\mathbf{z} = \pi_A(X') = V \operatorname{diag}(\mathbf{a}) \mathbf{1}_p / p = V \mathbf{a} / p \in \mathbb{R}^d. \tag{A35}$$

We also *simplify* CBAM by removing max-pooling from both attention mechanisms and keeping average pooling only. Then, (A32) takes the form

$$\begin{aligned} \mathbf{s} = \pi_A(V^\top) = V^\top \mathbf{1}_d / d &= (\operatorname{diag}(\mathbf{q}) X)^\top \mathbf{1}_d / d && \text{(A36)} \\ &= X^\top \mathbf{q} / d \in \mathbb{R}^p. && \text{(A37)} \end{aligned}$$

This reveals *pairwise interaction* by dot-product similarity between $\mathbf{q}$ as query and X as key. It was not evident in the original formulation, because dot product was split into element-wise product followed by sum.

We conclude that this modified CBAM module is a non-iterative instance of our pooling framework with $\mathbf{u}^0 = \pi_A(X) \in \mathbb{R}^d$, query mapping $\phi_Q(\mathbf{u}) = \sigma(\mathrm{MLP}(\mathbf{u}))/d \in \mathbb{R}^d$, key mapping $\phi_K = \mathrm{id}$, pairwise similarity function $s(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}$, spatial attention $\mathbf{a} = h(\mathbf{s}) = \sigma(\mathrm{conv}_7(\mathbf{s}))/p \in \mathbb{R}^p$, value mapping $\phi_V(X) = \operatorname{diag}(\mathbf{q}) X \in \mathbb{R}^{d \times p}$, average pooling function $f = f_{-1}$ and output mapping $\phi_U = \mathrm{id}$.

The original design does not use $\mathbf{z}$; instead, it has an output mapping $\phi_X(X) = V \operatorname{diag}(\mathbf{a}) = \operatorname{diag}(\mathbf{q}) X \operatorname{diag}(\mathbf{a}) \in \mathbb{R}^{d \times p}$. Thus, it can be used iteratively along with other mappings of X to form a modified network architecture.

SimPool is similar in that $\mathbf{u}^0 = \pi_A(X)$ but otherwise its attention mechanism is different: there is no channel attention while in spatial attention there are learnable query/key mappings and competition between spatial locations.

A2.5. Group 4: Transformers

We re-formulate the standard ViT [18] in two streams, where one performs pooling and the other feature mapping. We thus show that the pooling stream is an iterative instance of our framework, where iterations are blocks. We then examine the variant CaiT [80], which is closer to SimPool in that pooling takes place in the upper few layers with the features being fixed.

Vision transformer (ViT) [18] The transformer encoder *tokenizes* the input image, *i.e.*, it splits the image into p non-overlapping *patches* and maps them to patch token embeddings of dimension d through a linear mapping. It then concatenates a learnable CLS token embedding, also of dimension d, and adds a learnable *position embedding* of dimension d to all tokens. It is thus initialized as

$$F^0 = (\mathbf{u}^0 \;\; X^0) \in \mathbb{R}^{d \times (p+1)}, \tag{A38}$$

where $\mathbf{u}^0 \in \mathbb{R}^d$ is the initial CLS token embedding and $X^0 \in \mathbb{R}^{d \times p}$ contains the initial patch embeddings.

The encoder contains a sequence of *blocks*. Given token embeddings $F^t = (\mathbf{u}^t \;\; X^t) \in \mathbb{R}^{d \times (p+1)}$ as input, a block performs the following operations:

$$G^t = F^t + \mathrm{MSA}(\mathrm{LN}(F^t)) \in \mathbb{R}^{d \times (p+1)} \tag{A39}$$

$$F^{t+1} = G^t + \mathrm{MLP}(\mathrm{LN}(G^t)) \in \mathbb{R}^{d \times (p+1)}, \tag{A40}$$

where LN is LayerNorm [1] and MLP is a network of two affine layers with a ReLU activation in-between, applied to all tokens independently. Finally, at the end of block $T-1$, the image is pooled into vector $\mathbf{u} = \mathrm{LN}(\mathbf{u}^T)$.

Given $F^t \in \mathbb{R}^{d \times (p+1)}$, the *multi-head self-attention* (MSA) operation uses three linear mappings to form the query $Q = W_Q F^t$, key $K = W_K F^t$ and value $V = W_V F^t$, all in $\mathbb{R}^{d \times (p+1)}$. It then splits each of the three into m submatrices, each of size $d/m \times (p+1)$, where m is the number of *heads*.

Given a stacked matrix $A = (A_1; \dots; A_m) \in \mathbb{R}^{d \times n}$, where $A_i \in \mathbb{R}^{d/m \times n}$ for $i \in [m]$, we denote splitting as

$$\mathcal{A} = g_m(A) = \{A_1, \dots, A_m\} \subset \mathbb{R}^{d/m \times n}. \tag{A41}$$

Thus, with $\mathcal{Q} = g_m(Q) = \{Q_i\}$, $\mathcal{K} = g_m(K) = \{K_i\}$, $\mathcal{V} = g_m(V) = \{V_i\}$, self-attention is defined as

$$A_i = \boldsymbol{\sigma}_2 \left(K_i^\top Q_i / \sqrt{d'} \right) \in \mathbb{R}^{(p+1) \times (p+1)} \tag{A42}$$

$$Z_i = V_i A_i \in \mathbb{R}^{d' \times (p+1)}, \tag{A43}$$

for $i \in [m]$, where $d' = d/m$. Finally, given $\mathcal{Z} = \{Z_i\}$, submatrices are grouped back and an output linear mapping yields the output of MSA:

$$U = W_U g_m^{-1}(\mathcal{Z}) \in \mathbb{R}^{d \times (p+1)}. \tag{A44}$$

Here, we decompose the above formulation into two parallel streams. The first operates on the CLS token embedding $\mathbf{u}^t \in \mathbb{R}^d$, initialized by learnable parameter $\mathbf{u}^0 \in \mathbb{R}^d$ and iteratively performing pooling. The second operates on the patch embeddings $X^t \in \mathbb{R}^{d \times p}$, initialized by $X^0 \in \mathbb{R}^{d \times p}$ as obtained by tokenization and iteratively performing feature extraction. We focus on the first one.

Given $\mathbf{u}^t \in \mathbb{R}^d$, $X^t \in \mathbb{R}^{d \times p}$ at iteration t, we form the query $\mathcal{Q} = g_m(W_Q \mathrm{LN}(\mathbf{u}^t))$, key $\mathcal{K} = g_m(W_K \mathrm{LN}(X^t))$ and value $\mathcal{V} = g_m(W_V \mathrm{LN}(X^t))$. *Cross-attention* between $\mathcal{Q}$ and $\mathcal{K}, \mathcal{V}$ follows for $i \in [m]$:

$$\mathbf{a}_i = \boldsymbol{\sigma}_2 \left(K_i^\top \mathbf{q}_i / \sqrt{d'} \right) \in \mathbb{R}^p \tag{A45}$$

$$\mathbf{z}_i = V_i \mathbf{a}_i \in \mathbb{R}^{d'}. \tag{A46}$$

Finally, denoting $\mathcal{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_m\}$, the CLS token embedding at iteration $t+1$ is given by

$$\mathbf{g}^t = \mathbf{u}^t + W_U g_m^{-1}(\mathcal{Z}) \in \mathbb{R}^d \tag{A47}$$

$$\mathbf{u}^{t+1} = \mathbf{g}^t + \mathrm{MLP}(\mathrm{LN}(\mathbf{g}^t)) \in \mathbb{R}^d. \tag{A48}$$

We now simplify the above formulation by removing LayerNorm and residual connections. We also remove the dependence of self-attention of patch embeddings on the CLS token.

> We conclude that ViT [18] is an iterative instance of our pooling framework with learnable $\mathbf{u}^0 \in \mathbb{R}^d$, query mapping $\phi_Q(\mathbf{u}) = g_m(W_Q\mathbf{u}) \subset \mathbb{R}^{d'}$ with $d' = d/m$, key mapping $\phi_K(X) = g_m(W_K X) \subset \mathbb{R}^{d' \times p}$, pairwise similarity function $s(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}$, spatial attention $\mathcal{A} = h(\mathcal{S}) = \{\boldsymbol{\sigma}_2(\mathbf{s}_i/\sqrt{d'})\}_{i=1}^m \subset \mathbb{R}^p$, value mapping $\phi_V(X) = g_m(W_V X) \subset \mathbb{R}^{d' \times p}$, average pooling function $f = f_{-1}$ and output mappings $\phi_X(X) = \text{MLP}(\text{MSA}(X)) \in \mathbb{R}^{d \times p}$ and $\phi_U(\mathcal{Z}) = \text{MLP}(W_U g_m^{-1}(\mathcal{Z})) \in \mathbb{R}^d$.

Although $k = 1$, splitting into m submatrices and operating on them independently is the same as defining m query vectors in $\mathbb{R}^d$ via the block-diagonal matrix

$$Q = \begin{pmatrix} \mathbf{q}_1 & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{q}_m \end{pmatrix} \in \mathbb{R}^{d \times m}. \tag{A49}$$

Q interacts with K by dot product, essentially operating in m orthogonal subspaces. This gives rise to an attention matrix $A \in \mathbb{R}^{p \times m}$ containing $\mathbf{a}_i$ (A45) as columns and a pooled matrix $Z \in \mathbb{R}^{d \times m}$ containing $\mathbf{z}_i$ (A46) as columns.

Thus, the m heads in multi-head attention bear similarities to the k pooled vectors in our formulation. The fact that transformer blocks act as iterations strengthens our observation that methods with $k > 1$ are iterative. However, because of linear maps at every stage, there is no correspondence between heads across iterations.

Class-attention in image transformers (CaiT) [80] This work proposes two modifications in the architecture of ViT [18]. The first is that the encoder consists of two stages. In stage one, patch embeddings are processed alone, without a CLS token. In stage two, a learnable CLS token is introduced that interacts with patch embeddings with cross-attention, while the patch embeddings remain fixed. The second modification is that it introduces two learnable diagonal matrices $\Lambda_G^t, \Lambda_X^t \in \mathbb{R}^{d \times d}$ at each iteration (block) t and uses them to re-weight features along the channel dimension.

Thus, stage one is specified by a modification of (A39), (A40) as follows:

$$G^t = X^t + \Lambda_G^t \text{MSA}(\text{LN}(X^t)) \in \mathbb{R}^{d \times p} \tag{A50}$$

$$X^{t+1} = G^t + \Lambda_X^t \text{MLP}(\text{LN}(G^t)) \in \mathbb{R}^{d \times p}. \tag{A51}$$

This is similar to [29, 92], only here the parameters are learnable rather than obtained by GAP. Similarly, stage two is specified by a modification of (A45)-(A48). Typically, stage two consists only of a few (1-3) iterations.

> We conclude that a simplified version of stage two of CaiT [80] is an iterative instance of our pooling framework with the same options as ViT [18] except for the output mapping $\phi_X = \text{id}$.

SimPool is similar in that there are again two stages, but stage one is the entire encoder, while stage two is a single non-iterative cross-attention operation between features and their GAP, using function f_α for pooling.

Slot attention [49] is also similar to stage two of CaiT, performing few iterations of cross-attention between features and slots with $\phi_X = \text{id}$, but with a single head, $k > 1$ and different mapping functions.

Algorithm 2: SimPool. Green: learnable.

```
input : d: dimension, p: patches
input : features $X \in \mathbb{R}^{d \times p}$
output: pooled vector $\mathbf{u} \in \mathbb{R}^d$
$\mathbf{u}^0 \leftarrow X\mathbf{1}_p/p \in \mathbb{R}^d$                ▷ initialization (12)
$X \leftarrow \text{LN}(X) \in \mathbb{R}^{d \times p}$                ▷ LayerNorm [1]
$\mathbf{q} \leftarrow W_Q\mathbf{u}^0 \in \mathbb{R}^d$                ▷ query (13)
$K \leftarrow W_K X \in \mathbb{R}^{d \times p}$                ▷ key (14)
$\mathbf{a} \leftarrow \boldsymbol{\sigma}_2(K^\top\mathbf{q}/\sqrt{d}) \in \mathbb{R}^p$                ▷ attention (15)
$V \leftarrow X - \min X \in \mathbb{R}^{d \times p}$                ▷ value (16)
$\mathbf{u} \leftarrow f_\alpha^{-1}(f_\alpha(V)\mathbf{a}) \in \mathbb{R}^d$                ▷ pooling(8), (17)
```

A2.6. SimPool

SimPool is summarized in algorithm 2. We are given a *feature matrix* $X \in \mathbb{R}^{d \times p}$, resulting from flattening of tensor $\mathbf{X} \in \mathbb{R}^{d \times W \times H}$ into $p = W \times H$ patches. We form the initial representation $\mathbf{u}^0 = \pi_A(X) \in \mathbb{R}^d$ (12) by *global average pooling* (GAP), which is then mapped by $W_Q \in \mathbb{R}^{d \times d}$ (13) to form the *query* vector $\mathbf{q} \in \mathbb{R}^d$. After applying LayerNorm [1], $X' = \text{LN}(X)$, we map X' by $W_K \in \mathbb{R}^{d \times d}$ (14) to form the *key* $K \in \mathbb{R}^{d \times p}$. Then, $\mathbf{q}$ and K interact to generate the attention map $\mathbf{a} \in \mathbb{R}^p$ (15). Finally, the pooled representation $\mathbf{u} \in \mathbb{R}^d$ is a generalized weighted average of the *value* $V = X' - \min X' \in \mathbb{R}^{d \times p}$ with $\mathbf{a}$ determining the weights and scalar function f_α (8) determining the pooling operation (17).

The addition to what presented in the paper is LayerNorm after obtaining $\mathbf{u}^0$ and before K, V. That is, (14) and (16) are modified as

$$K = \phi_K(X) = W_K \text{LN}(X) \in \mathbb{R}^{d \times p}. \tag{A52}$$

$$V = \phi_V(X) = \text{LN}(X) - \min \text{LN}(X) \in \mathbb{R}^{d \times p}. \tag{A53}$$

As shown in Table 10, it is our choice in terms of simplicity, performance, and attention map quality to apply LayerNorm to key and value and linear layers to query and key. The learnable parameters are W_Q and W_K.

> In summary, SimPool is a non-iterative instance of our pooling framework with $k = 1$, $\mathbf{u}^0 = \pi_A(X) \in \mathbb{R}^d$, query mapping $\phi_Q(\mathbf{u}) = W_Q \mathbf{u} \in \mathbb{R}^d$, key mapping $\phi_K(X) = W_K \operatorname{LN}(X) \in \mathbb{R}^{d \times p}$, pairwise similarity function $s(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}$, spatial attention $\mathbf{a} = h(\mathbf{s}) = \boldsymbol{\sigma}_2(\mathbf{s}/\sqrt{d}) \in \mathbb{R}^p$, value mapping $\phi_V(X) = \operatorname{LN}(X) - \min \operatorname{LN}(X) \in \mathbb{R}^{d \times p}$, average pooling function $f = f_\alpha$ and output mapping $\phi_U = \text{id}$.

A3. More experiments

A3.1. More datasets, networks and protocols

Downstream tasks For *image classification*, we use CIFAR-10 [40], CIFAR-100 [40] and Oxford Flowers [58]. CIFAR-10 consists of 60,000 images in 10 classes, with 6,000 images per class. CIFAR-100 is just like CIFAR-10, except it has 100 classes containing 600 images each. Oxford Flowers consists of 102 flower categories containing between 40 and 258 images each.

For *semantic segmentation*, we fine-tune a linear layer of a self-supervised ViT-S on ADE20K [102], measuring mIoU, mAcc, and aAcc. The training set consists of 20k images and the validation set of 2k images in 150 classes.

For *background changes*, we use the linear head and linear probe of a supervised and self-supervised ViT-S, respectively, measuring top-1 classification accuracy on ImageNet-1k-9 [95] (IN-9) dataset. IN-9 contains nine coarse-grained classes with seven variations of both background and foreground.

For *image retrieval*, we extract features from a self-supervised ResNet-50 and ViT-S and evaluate them on $\mathcal{R}$Oxford and $\mathcal{R}$Paris [68], measuring mAP. These are the revisited Oxford [64] and Paris [65] datasets, comprising 5,062 and 6,412 images collected from Flickr by searching for Oxford and Paris landmarks respectively.

For *fine-grained classification*, we extract features from a supervised and self-supervised ResNet-50 and ViT-S and evaluate them on Caltech-UCSD Birds (CUB200) [86], Stanford Cars (CARS196) [39], In-Shop Clothing Retrieval (In-Shop) [46] and Stanford Online Products (SOP) [60], measuring Revall@k. Dataset statistics are summarized in Table A1.

For *unsupervised object discovery*, we use VOC07 [20] trainval, VOC12 [21] trainval and COCO 20K [44, 85]. The latter is a subset of COCO2014 trainval dataset [44], comprising 19,817 randomly selected images. VOC07 comprises 9,963 images depicting 24,640 annotated objects. VOC12 comprises 11,530 images depicting 27,450 annotated objects.

Dataset	CUB200	Cars196	SOP	In-Shop
Objects	birds	cars	furniture	clothes
# classes	200	196	22,634	7,982
# train images	5,894	8,092	60,026	26,356
# test images	5,894	8,093	60,027	26,356

Table A1. *Statistics and settings* for the four fine-grained classification datasets.

Ablation For the ablation of subsection A3.4, we train supervised ResNet-18 and ViT-T for *image classification* on ImageNet-20% and ImageNet-1k respectively.

A3.2. Implementation details

Analysis We train ResNet-18 on ImageNet-20% for 100 epochs following the ResNet-50 recipe of [91], but with learning rate 0.1. We train on 4 GPUs with a global batch size of $4 \times 128 = 512$, using SGD [71] with momentum. We incorporate pooling methods as a layer at the end of the model.

Group 1. For HOW [78], we use a kernel of size 3 and do not perform dimension reduction. For LSE [66], we initialize the scale as $r = 10$. For GeM [69], we use a kernel of size 7 and initialize the exponent as $p = 2$.

Group 2. For k-means, OTK [53] and slot attention [49], we set $k = 3$ vectors and take the maximum of the three logits per class. For convergence, we set tolerance $t = 0.01$ and iterations $T = 5$ for k-means. We set the iterations to $T = 3$ for OTK and slot attention.

Group 3. For CBAM [92], we use a kernel of size 7. For SE [29] and GE [28], we follow the implementation of [91].

Group 4. For ViT [18] and CaiT [80] we use $m = 4$ heads. For CaiT we set the iterations to $T = 1$, as this performs best.

Benchmark For *supervised* pre-training, we train ResNet-50 for 100 and 200 epochs, ConvNeXt-S and ViT-S for 100 and 300 epochs and ViT-B for 100 epochs on ImageNet-1k. For ResNet-50 we follow [91], using SGD with momentum with learning rate 0.4. We train on 8 GPUs with global batch size $8 \times 128 = 1024$. For ConvNeXt-S we follow [47], using AdamW [50] with learning rate 0.004. We use 8 GPUs with an aggregation factor of 4 (backpropagating every 4 iterations), thus with global batch size $8 \times 4 \times 256 = 4096$. For ViT-S we follow [91], using AdamW with learning rate 5×10^{-4}. We train on 8 GPUs with global batch size $8 \times 74 = 592$. For the 300 epoch experiments, we follow the same setup as for 100.

For *self-supervised* pre-training, we train ResNet-50, ConvNeXt-S and ViT-S with DINO [8] on ImageNet-1k for 100 and 300 epochs, following [8] and using 6 local crops.

For ResNet-50, we train on 8 GPUs with global batch size $8 \times 160 = 1280$. We use learning rate 0.3, minimum learning rate 0.0048, global crop scale $[0.14, 1.0]$ and local crop scale $[0.05, 0.14]$. For ConvNeXt-S, we train on 8 GPUs with global batch size $8 \times 60 = 480$. We use learning rate 0.001, minimum learning rate 2×10^{-6}, global crop scale $[0.14, 1.0]$ and local crop scale $[0.05, 0.14]$. As far as we know, we are the first to integrate DINO into ConvNeXt-S. For ViT-S, we train on 8 GPUs with global batch size $8 \times 100 = 800$. We use LARS [96] with learning rate 5×10^{-4}, minimum learning rate of 1×10^{-5}, global crop scale $[0.25, 1.0]$ and local crop scale $[0.05, 0.25]$. For the 300 epoch experiments, we follow the same setup as for 100. For linear probing, we follow [8], using 4 GPUs with global batch size $4 \times 256 = 1024$.

Downstream tasks For *image classification*, we fine-tune supervised and self-supervised ViT-S on CIFAR-10, CIFAR-100 and Oxford Flowers, following [103]. We use a learning of 7.5×10^{-6}. We train on 8 GPUS for 1000 epochs with a global batch size of $8 \times 96 = 768$.

For *object localization*, we use the supervised and self-supervised ViT-S on CUB and ImageNet-1k, without fine-tuning. We follow [11] and we use the MaxBoxAccV2 metric. For the baseline, we use the mean attention map over all heads of the CLS token to generate the bounding boxes. For SimPool, we use the attention map $\mathbf{a}$ (15).

For *unsupervised object discovery*, we use the self-supervised ViT-S on VOC07 [20] trainval, VOC12 [21] trainval and COCO 20K [44, 85], without fine-tuning. We adopt LOST [73] and DINO-seg [73, 8] to extract bounding boxes. For both methods, we follow the best default choices [73]. LOST operates on features. We use the the *keys* of the last self-attention layer for the baseline and the *keys* K (14) for SimPool. DINO-seg operates on attention maps. We use the attention map of the head that achieves the best results following [73], *i.e.* head 4, for the baseline and the attention map $\mathbf{a}$ (15) for SimPool.

For *semantic segmentation*, we use the self-supervised ViT-S on ADE20K [102]. To evaluate the quality of the learned representation, we only fine-tune a linear layer on top of the fixed patch features, without multi-scale training or testing and with the same hyper-parameters as in iBOT [103]. We follow the setup of [45], *i.e.*, we train for 160,000 iterations with 512×512 images. We use AdamW [50] optimizer with initial learning rate 3×10^{-5}, poly-scheduling and weight decay of 0.05. We train on 4 GPUS with a global batch size of $4 \times 4 = 16$.

For *computation resources*, we measure GFLOPS for input size 224×224 on a single NVIDIA A100 40GB GPU.

A3.3. More benchmarks

Self-supervised pre-training On the 100% of ImageNet-1k, we train ViT-S with DINO [8] for 300 epochs. Table A2 shows that SimPool improves over the baseline by 0.4% k-NN and 0.7% linear probing.

Method	Epochs	ViT-S	
		k-NN	Prob
Baseline	300	72.2	74.3
SimPool	300	**72.6**	**75.0**

Table A2. *Image classification* top-1 accuracy (%) on ImageNet-1k. Self-supervised pre-training with DINO [8] for 300 epochs. Baseline: GAP for convolutional, CLS for transformers.

Method	mIoU	mAcc	aAcc
Baseline	26.4	34.0	71.6
SimPool	**27.9**	**35.7**	**72.6**

Table A3. *Semantic segmentation* on ADE20K [102]. ViT-S pre-trained on ImageNet-1k for 100 epochs. Self-supervision with DINO [8].

Network	Method	$\mathcal{R}$Oxford		$\mathcal{R}$Paris	
		Medium	Hard	Medium	Hard
ResNet-50	Baseline	27.2	7.9	47.3	19.0
	SimPool	**29.7**	**8.7**	**51.6**	**23.0**
ViT-S	Baseline	29.4	10.0	54.6	26.2
	SimPool	**32.1**	**10.6**	**56.5**	**27.3**

Table A4. *Image retrieval* mAP (%) without fine-tuning on $\mathcal{R}$Oxford and $\mathcal{R}$Paris [68]. Self-supervised pre-training with DINO [8] on ImageNet-1k for 100 epochs.

Semantic segmentation We evaluate semantic segmentation on ADE20K [102] under self-supervised pre-training. To evaluate the quality of the learned representation, we only fine-tune a linear layer on top of the fixed patch features, as in iBOT [103]. Table A3 shows that SimPool increases all scores by more than 1% over the baseline. These results testify the improved quality of the learned representations when pre-training with SimPool.

Background changes Deep neural networks often rely on the image background, which can limit their ability to generalize well. To achieve better performance, these models must be able to cope with changes in the background and prioritize the foreground. To evaluate SimPool robustness to the background changes, we use the ImageNet-1k-9 [95] (IN-9) dataset. In four of these datasets, *i.e.*, Only-FG (OF), Mixed-Same (MS), Mixed-Rand (MR), and Mixed-Next (MN), the background is modified. The three other datasets feature masked foregrounds, *i.e.*, No-FG (NF), Only-BG-B (OBB), and Only-BG-T (OBT).

Image retrieval without fine-tuning While classification accuracy indicates ability of a model to recognize objects of the same classes as those it was trained for, it does not nec-

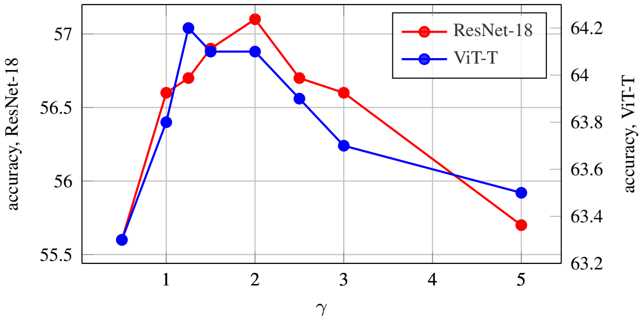

Figure A1. *Image classification* top-1 accuracy (%) *vs. exponent* $\gamma = (1 - \alpha)/2$ (17) for ResNet-18 supervised on ImageNet-20% and ViT-T supervised on ImageNet-1k, both for 100 epochs.

essarily reflect its ability to capture the visual similarity between images, when tested on a dataset from a different distribution. Here, we evaluate this property of visual features using ResNet-50 and ViT-S; for particular object retrieval without fine-tuning on $\mathcal{R}$Oxford and $\mathcal{R}$Paris [68]. In Table A4, we observe that SimPool is very effective, improving the retrieval performance of both models on all datasets and evaluation protocols over the baseline.

Fine-grained classification We evaluate fine-grained classification using ResNet-50 and ViT-S, both supervised and self-supervised, following [36]. We extract features from test set images and directly apply nearest neighbor search, measuring Recall@k. Table A5 shows that SimPool is superior to the baseline in most of the datasets, models and supervision settings, with the exception of ResNet-50 supervised on In-Shop, ResNet-50 self-supervised on Cars196 and ViT-S self-supervised on SOP (3 out of 16 cases). The improvement is roughly 1-2% Recall@1 in most cases, and is most pronounced on self-supervised on CUB200, roughly 5%.

A3.4. More ablations

Pooling parameter α (17) We ablate the effect of parameter α of the pooling function f_α (17) on the classification performance of SimPool using ResNet-18 on ImageNet-20% and ViT-T on ImageNet-1k for 100 epochs. We find learnable α (or $\gamma = (1 - \alpha)/2$) to be inferior both in terms of performance and attention map quality. For ResNet-18 on ImageNet-20%, it gives top-1 accuracy 56.0%. Clamping to $\gamma = 5$ gives 56.3% and using a $10\times$ smaller learning rate gives 56.5%.

In Figure A1, we set exponent γ to be a hyperparameter and observe that for both networks, values between 1 and 3 are relatively stable. Specifically, the best choice is 2 for ResNet-18 and 1.25 for ViT-T. Thus, we choose exponent 2 for convolutional networks (ResNet-18, ResNet-50 and ConvNeXt-S) and 1.25 for vision transformers (ViT-T, ViT-S and ViT-B).

A3.5. More visualizations

Attention maps: ViT Figure A2 shows attention maps of supervised and self-supervised ViT-S trained on ImageNet-1k. The ViT-S baseline uses the CLS token for pooling by default. For SimPool, we remove the CLS stream entirely from the encoder and use the attention map $\mathbf{a}$ (15).

We observe that under *self-supervision*, the attention map quality of SimPool is on par with the baseline and in some cases the object of interest is slightly more pronounced, *e.g.*, rows 1, 3, 6 and 7.

What is more impressive is *supervised* training. In this case, the baseline has very low quality of attention maps, focusing only on part of the object of interest (*e.g.*, rows 1, 2, 5, 6, 10), focusing on background more than self-supervised (*e.g.*, rows 1, 4, 6, 7, 8), even missing the object of interest entirely (*e.g.*, rows 3, 9). By contrast, the quality of attention maps of SimPool is superior even to self-supervised, attending more to the object surface and less background.

Segmentation masks Figure A3 shows the same images for the same setting as in Figure A2, but this time overlays segmenation masks on top input images, corresponding to more than 60% mass of the attention map. Again, SimPool is on par with baseline when self-supervised, supervised baseline has poor quality and supervised SimPool is a lot better, although its superiority is not as evident as with the raw attention maps.

Object localization Figure A4 visualizes object localization results, comparing bounding boxes of SimPool with the baseline. The results are obtained from the experiments of Table 5, using ViT-S with supervised pre-training. We observe that the baseline systematically fails to localize the objects accurately. On the other hand, SimPool allows reasonable localization of the object of interest just from the attention map, without any supervision other than the image-level label.

Attention maps: The effect of γ Figure A5 and Figure A6 visualize the effect of exponent $\gamma = (1 - \alpha)/2$ of pooling operation f_α (8) on the quality of the attention maps of ResNet-18 and ViT-T, respectively. The use of the average pooling operation f_{-1} as opposed to f_α (8) is referred to as no γ. For ResNet-18, we observe that for $\gamma < 1.25$ or $\gamma > 3.0$, the attention maps are of low quality, failing to delineate the object of interest (*e.g.*, rows 4, 5, 11), missing the object of interest partially (*e.g.*, rows 1, 2, 3, 6) or even entirely (*e.g.*, row 7). For ViT-T, it is impressive that for γ around or equal to 1.25, the attention map quality is high, attending more (*e.g.*, rows 1, 2, 4, 7) or even exclusively (*e.g.*, rows 3, 6, 11) the object instead of background.

Attention maps: CLS vs. SimPool Figure A7 compares the quality of the attention maps of supervised ViT-T trained with CLS to that of SimPool. For CLS, we visualize the

NETWORK	METHOD	CUB200			CARS196			SOP			IN-SHOP		
		R@1	2	4	R@1	2	4	R@1	10	100	R@1	10	20
SUPERVISED													
ResNet-50	Baseline	42.7	55.2	67.7	42.3	54.2	65.7	48.3	63.2	71.8	**27.6**	**49.9**	**56.5**
	SimPool	**43.0**	**55.2**	**67.9**	**43.8**	**56.2**	**67.4**	**48.7**	**64.1**	**72.9**	27.0	49.9	56.5
ViT-S	Baseline	55.8	68.3	78.3	38.2	50.3	61.8	54.1	69.2	81.6	30.9	56.5	63.2
	SimPool	**56.8**	**69.6**	**79.2**	**38.9**	**50.7**	**63.3**	**54.2**	**69.4**	**81.9**	**32.8**	**57.6**	**64.3**
SELF-SUPERVISED													
ResNet-50	Baseline	26.0	36.2	46.9	**34.1**	**44.2**	**55.0**	51.2	65.3	76.5	37.1	58.4	64.1
	SimPool	**30.7**	**40.9**	**53.3**	33.6	43.6	54.3	**52.1**	**66.5**	**77.2**	**38.1**	**60.0**	**65.6**
ViT-S	Baseline	56.7	69.4	80.5	37.5	47.5	58.4	**59.8**	**74.4**	**85.4**	40.4	63.9	70.3
	SimPool	**61.8**	**74.4**	**83.6**	**37.6**	**48.0**	**58.4**	59.5	73.9	85.0	**41.1**	**64.3**	**70.8**

Table A5. *Fine-grained classification* Recall@k (R@k, %) without fine-tuning on four datasets, following the same protocol as [55, 36]. Models pre-trained on ImageNet-1k for 100 epochs. Self-supervision with DINO [8].

mean attention map of the heads of the CLS token for each of the 12 blocks. For SimPool, we visualize the attention map $\mathbf{a}$ (15). SimPool has attention maps of consistently higher quality, delineating and exclusively focusing on the object of interest (*e.g.*, rows 6, 10, 13). It is impressive that while CLS interacts with patch tokens in 12 different blocks, it is inferior to SimPool, which interacts only once at the end.

Attention maps: ResNet, ConvNeXt Figure A8 and Figure A9 show attention maps of supervised and self-supervised ResNet-50 and ConvNeXt-S, respectively. Both networks are pre-trained on ImageNet-1k for 100 epochs. We use the attention map $\mathbf{a}$ (15). We observe that SimPool enables the default ResNet-50 and ConvNeXt-S to obtain raw attention maps of high quality, focusing on the object of interest and not on background or other objects. This is not possible with the default global average pooling and is a property commonly thought of vision transformers when self-supervised [8]. Between supervised and self-supervised SimPool, the quality differences are small, with self-supervised being slightly superior.

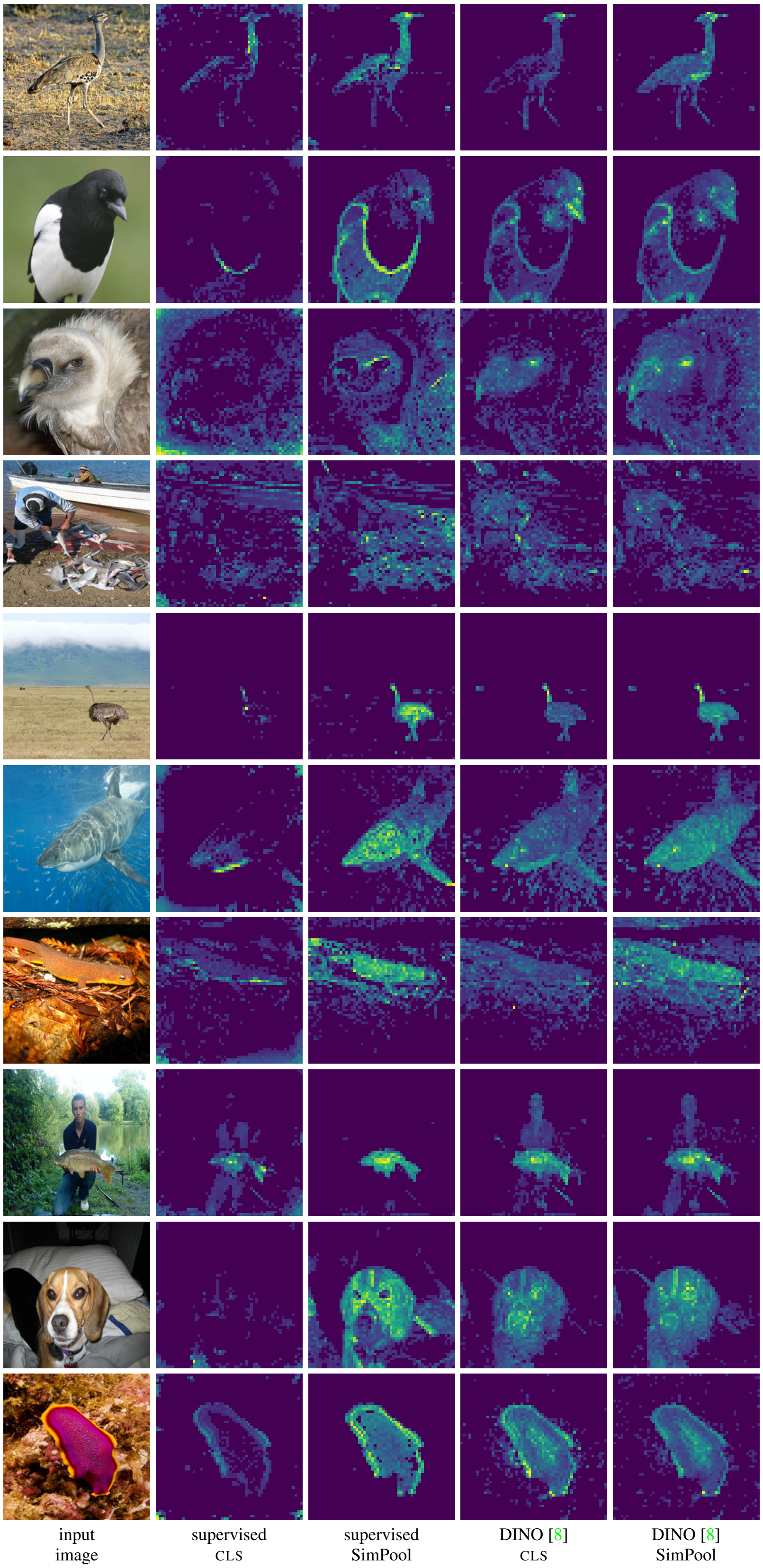

Figure A2. *Attention maps* of ViT-S [18] trained on ImageNet-1k for 100 epochs under supervision and self-supervision with DINO [8]. For ViT-S baseline, we use the mean attention map of the CLS token. For SimPool, we use the attention map $\mathbf{a}$ (15). Input image resolution: 896×896; patches: 16×16; output attention map: 56×56.

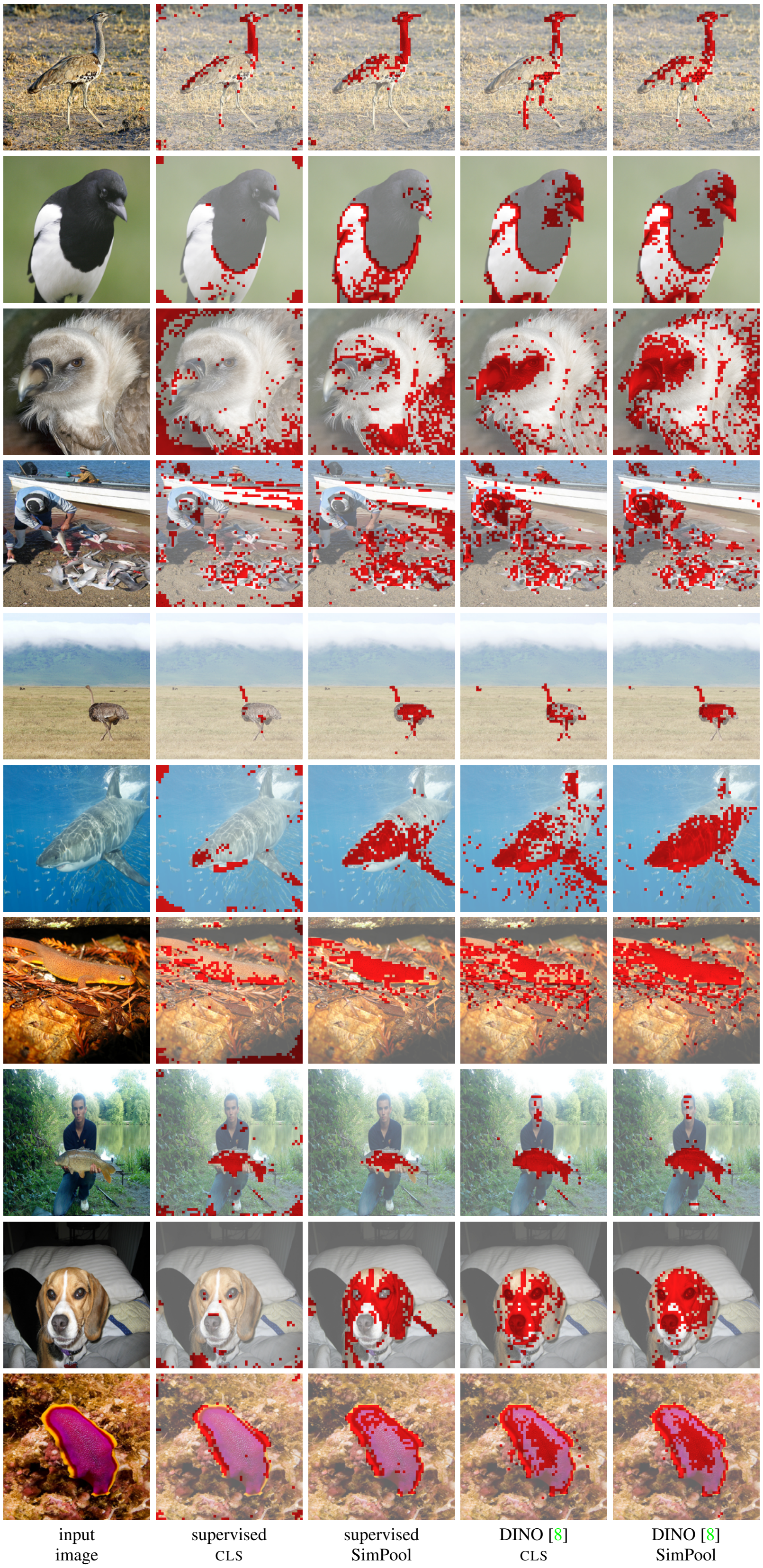

Figure A3. *Segmentation masks* of ViT-S [18] trained on ImageNet-1k for 100 epochs under supervision and self-supervision with DINO [8]. For ViT-S baseline, we use the attention map of the CLS token. For SimPool, we use the attention map $\mathbf{a}$ (15). Same as Figure A2, with attention map value thresholded at 60% of mass and mask overlaid on input image.

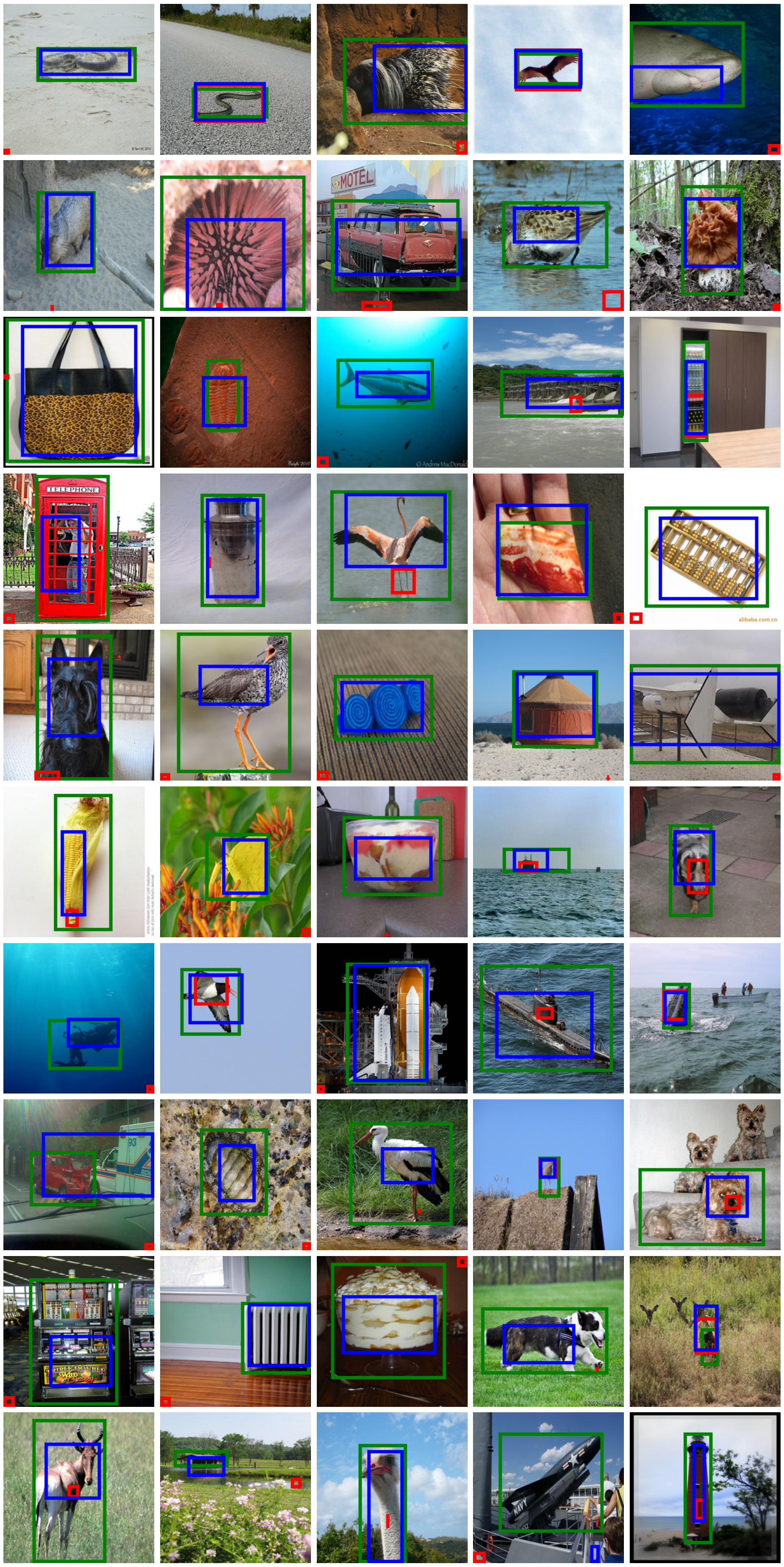

Figure A4. *Object localization* on ImageNet-1k with ViT-S [18] supervised pre-training on ImageNet-1k-1k for 100 epochs. Bounding boxes obtained from experiment of Table 5, following [11]. Green: ground-truth bounding boxes; red: baseline, predicted by the attention map of the CLS token; blue: predicted by SimPool, using the attention map $\mathbf{a}$ (15).

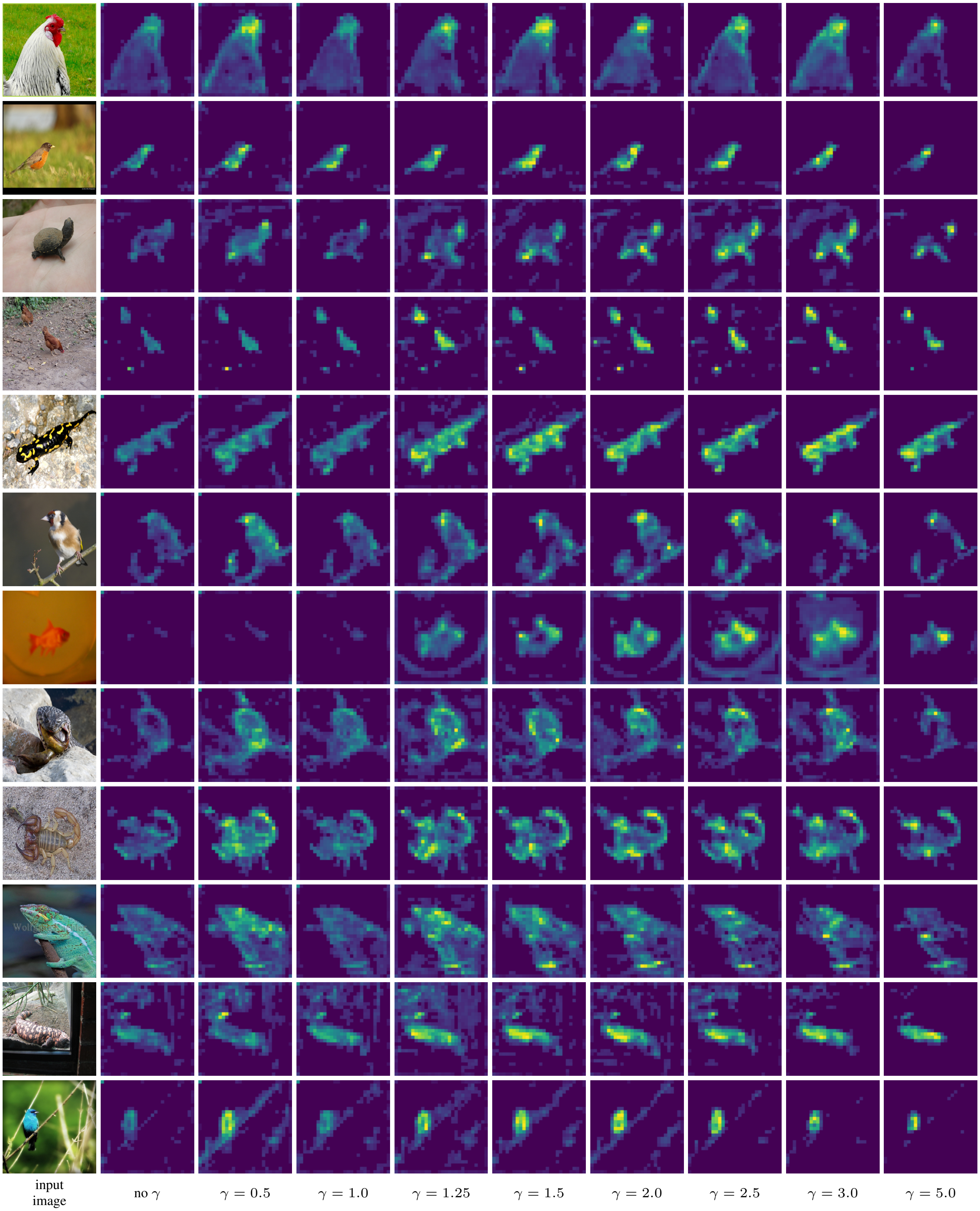

Figure A5. *The effect of* γ. Attention maps of ResNet-18 [26] with SimPool using different values of γ trained on ImageNet-20% for 100 epochs under supervision. We use the attention map $\mathbf{a}$ (15). Input image resolution: 896×896; output attention map: 28×28; no γ: using the average pooling operation f_{-1} instead of f_α (8). We set $\gamma = 2$ by default for convolutional networks.

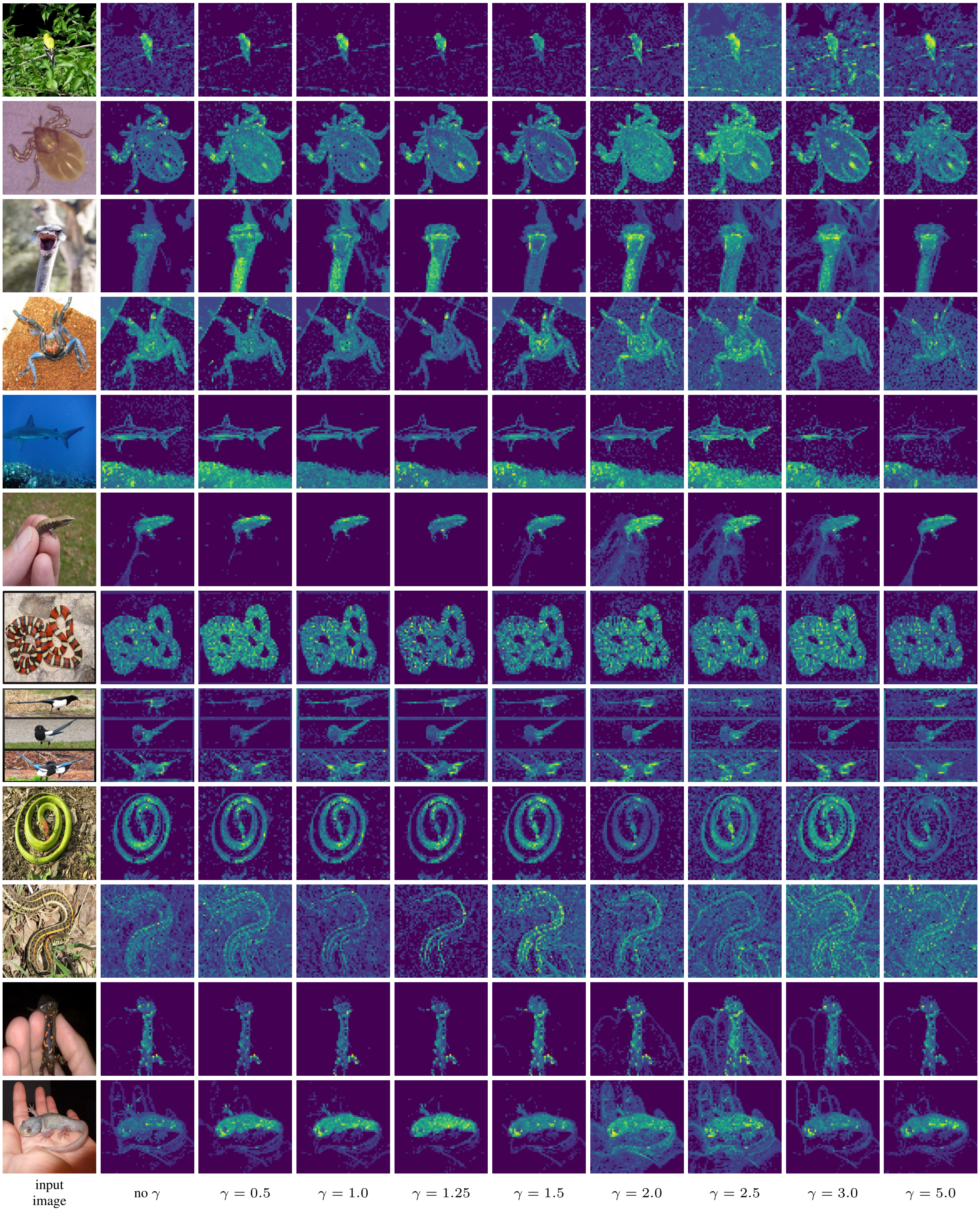

Figure A6. *The effect of* γ. Attention maps of ViT-T [18] with SimPool using different values of γ trained on ImageNet-1k for 100 epochs under supervision. We use the attention map $\mathbf{a}$ (15). Input image resolution: 896×896; patches: 16×16; output attention map: 56×56; no γ: using the average pooling operation f_{-1} instead of f_α (8). We set $\gamma = 1.25$ by default for transformers.

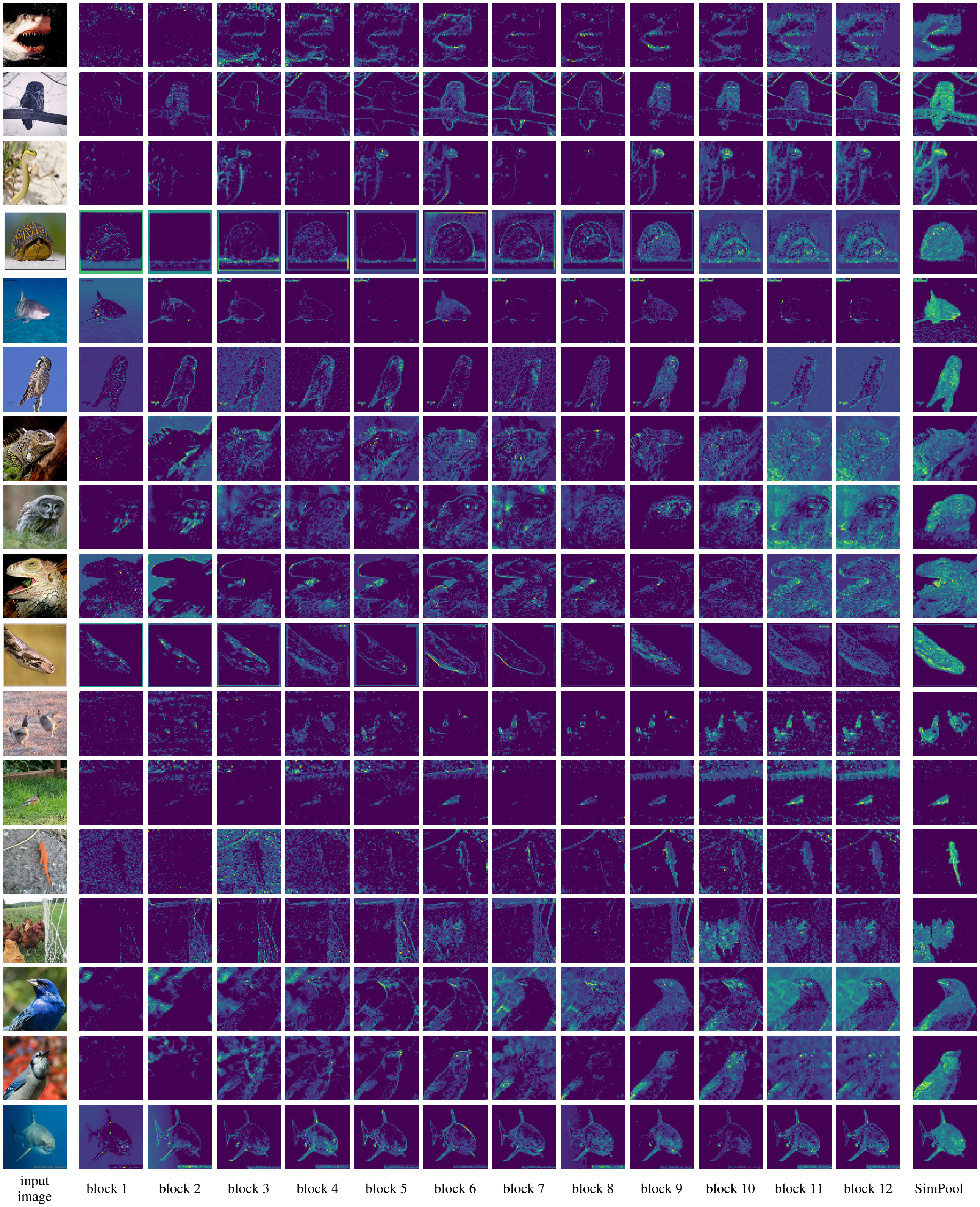

Figure A7. CLS *vs. SimPool*. Attention maps of ViT-T [18] trained on ImageNet-1k for 100 epochs under supervision. For CLS, we use the mean attention map of the CLS token of each block. For SimPool, we use the attention map $\mathbf{a}$ (15). Input image resolution: 896 × 896; patches: 16 × 16; output attention map: 56 × 56.

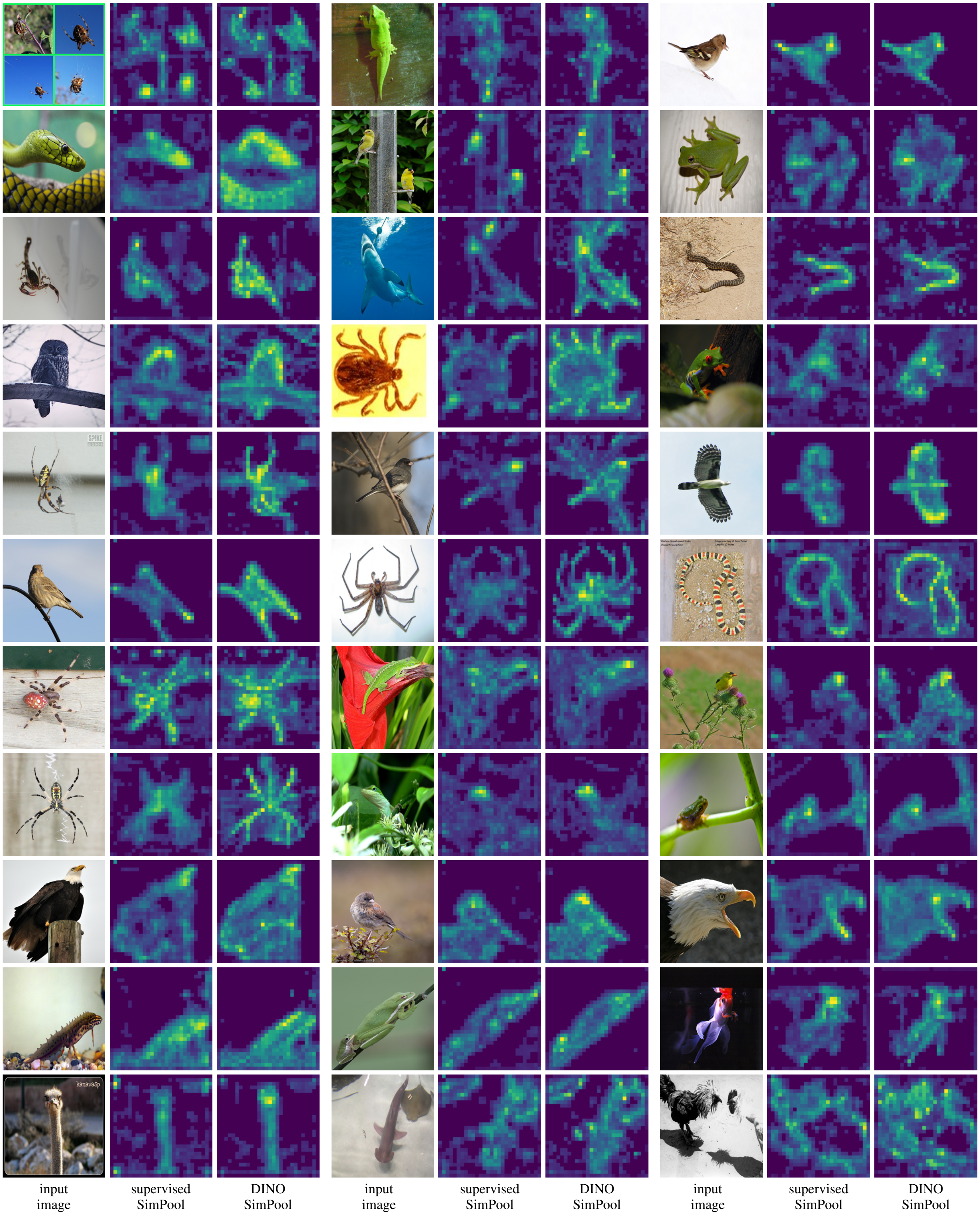

Figure A8. *Attention maps* of ResNet-50 [26] trained on ImageNet-1k for 100 epochs under supervision and self-supervision with DINO [8]. We use the attention map $\mathbf{a}$ (15). Input image resolution: 896 × 896; output attention map: 28 × 28.

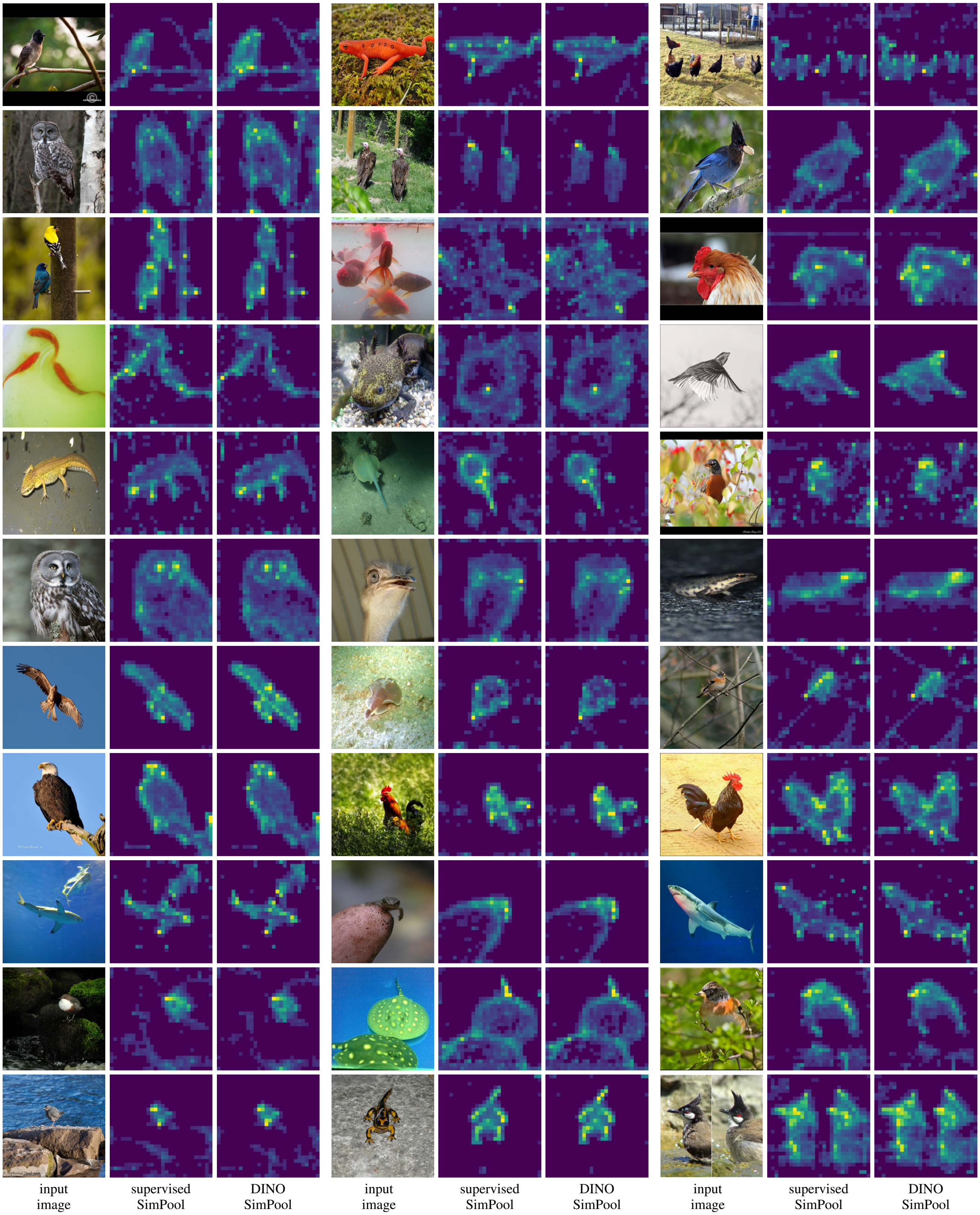

Figure A9. *Attention maps* of ConvNeXt-S [47] trained on ImageNet-1k for 100 epochs under supervision and self-supervision with DINO [8]. We use the attention map $\mathbf{a}$ (15). Input image resolution: 896 × 896; output attention map: 28 × 28.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 4, 6, 7

[2] Artem Babenko and Victor Lempitsky. Aggregating local deep features for image retrieval. In *International Conference on Computer Vision*, 2015. 1

[3] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3286–3295, 2019. 2

[4] Liefeng Bo and Cristian Sminchisescu. Efficient match kernel between sets of features for visual recognition. In *Advances in Neural Information Processing Systems 22*. 2009. 1

[5] Y-Lan Boureau, Francis Bach, Yann Lecun, and Jean Ponce. Learning mid-level features for recognition. In *Computer Vision and Pattern Recognition*, 2010. 1

[6] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118, 2010. 1

[7] Bingyi Cao, Andre Araujo, and Jack Sim. Unifying deep local and global features for image search. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, pages 726–743. Springer, 2020. 1

[8] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 2, 8, 9, 11, 12, 13, 18, 19

[9] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. Aˆ 2-nets: Double attention networks. *Advances in neural information processing systems*, 31, 2018. 2

[10] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Empirical Methods in Natural Language Processing*, 2014. 5

[11] Junsuk Choe, Seong Joon Oh, Seungho Lee, Sanghyuk Chun, Zeynep Akata, and Hyunjung Shim. Evaluating weakly supervised object localization methods right. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3133–3142, 2020. 9, 14

[12] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV Workshop on Statistical Learning in Computer Vision*, 2004. 1

[13] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013. 4

[14] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition*, volume 1, 2005. 1

[15] John G Daugman. Two-dimensional spectral analysis of cortical receptive field profiles. *Vision research*, 20(10):847–856, 1980. 1

[16] John G Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *Journal of Optical Society of America*, 2(7):1160–1169, 1985. 1

[17] John G Daugman. Complete discrete 2-d gabor transforms by neural networks for image analysis and compression. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(7):1169–1179, 1988. 1

[18] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 2, 6, 7, 8, 12, 13, 14, 16, 17

[19] Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning*, pages 2286–2296. PMLR, 2021. 2

[20] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. 8, 9

[21] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. 8, 9

[22] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980. 1

[23] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision*, 2015. 1

[24] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. LeViT: a vision transformer in convnet's clothing for faster inference. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2021. 2

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015. 1

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016. 1, 3, 15, 18

[27] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 2

[28] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Andrea Vedaldi. Gather-excite: Exploiting feature context in convolutional neural networks. *Advances in neural information processing systems*, 31, 2018. 1, 8

[29] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 5, 6, 7, 8

[30] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, Teddy Furon, and Ondrej Chum. Efficient diffusion on region manifolds: Recovering small objects with compact cnn representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017. 2

[31] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 2008. 1

[32] Herve Jegou, Matthijs Douze, and Cordelia Schmid. On the burstiness of visual elements. In *Computer Vision and Pattern Recognition*, 2009. 1

[33] Herve Jegou and Andrew Zisserman. Triangulation embedding and democratic aggregation for image search. In *Computer Vision and Pattern Recognition*, 2014. 1

[34] Y. Jia, C. Huang, and T. Darrell. Beyond spatial pyramids: Receptive field learning for pooled image features. In *Computer Vision and Pattern Recognition*, 2012. 1

[35] Hervé Jégou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Pérez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1704–1716, 2012. 1

[36] Ioannis Kakogeorgiou, Spyros Gidaris, Bill Psomas, Yannis Avrithis, Andrei Bursuc, Konstantinos Karantzalos, and Nikos Komodakis. What to hide from your students: Attention-guided masked image modeling. In *European Conference on Computer Vision*, 2022. 10, 11

[37] Yannis Kalantidis, Clayton Mellina, and Simon Osindero. Cross-dimensional weighting for aggregated deep convolutional features. In *European Conference on Computer Vision Workshops*, 2016. 1

[38] Philip A Knight. The Sinkhorn-Knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 2008. 4

[39] Jonathan Krause, Michael Stark, Jia Deng, and Fei-Fei Li. 3d object representations for fine-grained categorization. *ICCVW*, 2013. 8

[40] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 8

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*. 2012. 1

[42] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 1989. 1

[43] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013. 1, 3

[44] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014. 8, 9

[45] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 2, 9

[46] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 8

[47] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 1, 8, 19

[48] Stuart Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. 3, 4

[49] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 33, 2020. 3, 4, 5, 7, 8

[50] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018. 8, 9

[51] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004. 1

[52] J. Malik, S. Belongie, J. Shi, and T. Leung. Textons, contours and regions: Cue integration in image segmentation. In *Proceedings of the International Conference on Computer Vision*, volume 2, page 918, Jan 1999. 1

[53] Grégoire Mialon, Dexiong Chen, Alexandre d'Aspremont, and Julien Mairal. A trainable optimal transport embedding for feature aggregation and its relationship to attention. *arXiv preprint arXiv:2006.12065*, 2020. 2, 3, 4, 8

[54] Naila Murray and Florent Perronnin. Generalized max pooling. In *Computer Vision and Pattern Recognition*, 2014. 1

[55] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. In *European Conference on Computer Vision*, 2020. 11

[56] Muhammad Muzammal Naseer, Kanchana Ranasinghe, Salman H Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Intriguing properties of vision transformers. *Advances in Neural Information Processing Systems*, 34:23296–23308, 2021. 2

[57] Tony Ng, Vassileios Balntas, Yurun Tian, and Krystian Mikolajczyk. Solar: second-order loss and attention for image retrieval. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16*, pages 253–270. Springer, 2020. 1

[58] M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008. 8

[59] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. Large-scale image retrieval with attentive deep local features. In *Proceedings of the IEEE international conference on computer vision*, pages 3456–3465, 2017. 1

[60] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 8

[61] A. Oliva and A. Torralba. Building the gist of a scene: the role of global image features in recognition. *Visual Perception*, 2006. 1

[62] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition*, 2006. 1

[63] Elia Peruzzo, Enver Sangineto, Yahui Liu, Marco De Nadai, Wei Bi, Bruno Lepri, and Nicu Sebe. Spatial entropy regularization for vision transformers. *arXiv preprint arXiv:2206.04636*, 2022. 2

[64] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007. 8

[65] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 8

[66] Pedro O Pinheiro and Ronan Collobert. From image-level to pixel-level labeling with convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1713–1721, 2015. 3, 8

[67] Tobias Plötz and Stefan Roth. Neural nearest neighbors networks. *Advances in Neural information processing systems*, 31, 2018. 2

[68] Filip Radenović, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5706–5715, 2018. 8, 9, 10

[69] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-Tuning CNN Image Retrieval with No Human Annotation. *IEEE transactions on pattern analysis and machine intelligence*, 41(7):1655–1668, 2018. 1, 3, 8

[70] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34:12116–12128, 2021. 2

[71] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. 8

[72] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *Computer Vision and Pattern Recognition*, 2005. 1

[73] Oriane Siméoni, Gilles Puy, Huy V Vo, Simon Roburin, Spyros Gidaris, Andrei Bursuc, Patrick Pérez, Renaud Marlet, and Jean Ponce. Localizing objects with self-supervised transformers and no labels. In *BMVC-British Machine Vision Conference*, 2021. 9

[74] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 1

[75] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *International Conference Computer Vision*, volume 2, 2003. 1

[76] Sainbayar Sukhbaatar, Takaki Makino, and Kazuyuki Aihara. Auto-pooling: Learning to improve invariance of image features from image sequences. *arXiv preprint arXiv:1301.3323*, 2013. 1

[77] Giorgos Tolias, Yannis Avrithis, and Hervé Jégou. To aggregate or not to aggregate: Selective match kernels for image search. In *2013 IEEE International Conference on Computer Vision*, pages 1401–1408, 2013. 1

[78] Giorgos Tolias, Tomas Jenicek, and Ondřej Chum. Learning and aggregating deep local descriptors for instance-level recognition. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 460–477. Springer, 2020. 1, 3, 8

[79] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of CNN activations. In *4th International Conference on Learning Representations*, 2016. 1, 3

[80] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 32–42. IEEE, 2021. 2, 6, 7, 8

[81] Mark R Turner. Texture discrimination by gabor functions. *Biological cybernetics*, 55(2-3):71–82, 1986. 1

[82] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12894–12904, 2021. 2

[83] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017. 2

[84] Shashanka Venkataramanan, Amir Ghodrati, Yuki M Asano, Fatih Porikli, and Amirhossein Habibian. Skip-attention: Improving vision transformers by paying less attention. *arXiv preprint arXiv:2301.02240*, 2023. 2

[85] Huy V. Vo, Patrick Pérez, and Jean Ponce. Toward unsupervised, multi-object discovery in large-scale image collec-

tions. In *European Conference on Computer Vision*, 2020. 8, 9

[86] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 8

[87] Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks. In *CVPR*, 2020. 2

[88] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 568–578, 2021. 2

[89] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvt v2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 8(3):415–424, 2022. 2

[90] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 2

[91] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019. 8

[92] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 1, 5, 7, 8

[93] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. CvT: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 2

[94] Kan Wu, Houwen Peng, Minghao Chen, Jianlong Fu, and Hongyang Chao. Rethinking and improving relative position encoding for vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 2

[95] Kai Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. Noise or signal: The role of image backgrounds in object recognition. In *International Conference on Learning Representations*, 2021. 8, 9

[96] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017. 9

[97] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10819–10829, 2022. 2

[98] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, 2017. 1

[99] Qinglong Zhang and Yu-Bin Yang. Rest: An efficient transformer for visual recognition. *Advances in Neural Information Processing Systems*, 34:15475–15485, 2021. 2

[100] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10076–10085, 2020. 2

[101] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Computer Vision and Pattern Recognition*, June 2016. 1

[102] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 633–641, 2017. 8, 9

[103] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. In *International Conference on Learning Representations*, 2022. 9