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

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)

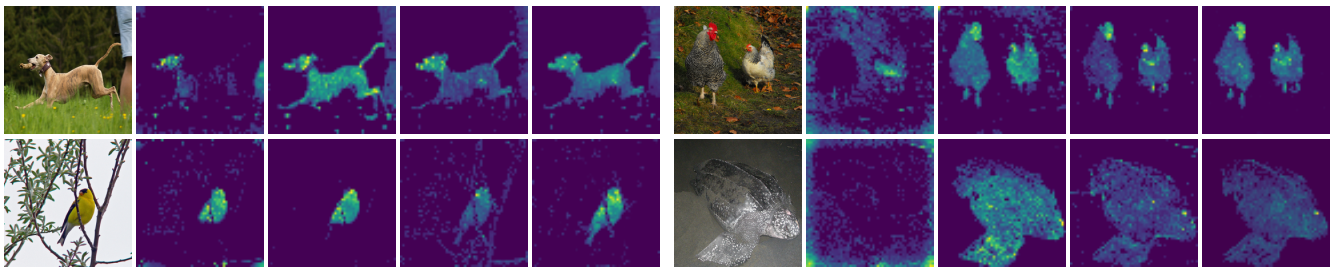

Figure 1. We introduce SimPool, a simple attention-based pooling method at the end of network, obtaining clean attention maps under supervision or self-supervision. Attention maps of ViT-S [14] trained on ImageNet-1k [11]. For baseline, we use the mean attention map of the CLS token. For SimPool, we use the attention map $\mathbf{a}$ (15). Input image: 896×896; patches: 16×16; attention map: 56×56.

Abstract

Convolutional networks and vision transformers have different forms of pairwise interactions, pooling across layers and pooling at the end of the network. Does the latter really need to be different? As a by-product of pooling, vision transformers provide spatial attention for free, but this is most often of low quality unless self-supervised, which is not well studied. Is supervision really the problem?

In this work, we develop a generic pooling framework and then we formulate a number of existing methods as instantiations. By discussing the properties of each group of methods, we derive SimPool, a simple attention-based pooling mechanism as a replacement of the default one for both convolutional and transformer encoders. We find that, whether supervised or self-supervised, this improves performance on pre-training and downstream tasks and provides attention maps delineating object boundaries in all cases. One could thus call SimPool universal. To our knowledge, we are the first to obtain attention maps in supervised transformers of at least as good quality as self-supervised, without explicit losses or modifying the architecture. Code at: https://github.com/billpsomas/simpool.

1. Introduction

Extracting visual representations and spatial pooling have been two interconnected processes since the study of 2D Gabor filters [10] and early convolutional networks [17]. Modern *convolutional networks* [20, 32] gradually perform local pooling and downsampling throughout the architecture to extract a low-resolution feature tensor, followed by global spatial pooling. *Vision transformers* [14] only downsample at input tokenization and then preserve resolution, but pooling takes place again throughout the architecture via the interaction of patch tokens with a CLS token, inherited from language models [13].

The pooling operation has been studied extensively in instance-level tasks on convolutional networks [3, 42], but less so in category-level tasks or transformers. Pooling in transformers is based on weighted averaging, using as weights the 2D *attention map* of the CLS token at the last layer. However, this attention map is typically of low quality, unless under self-supervision [7].

In this work, we argue that vision transformers can be reformulated in two streams, where one is extracting a visual representation on patch tokens and the other is performing spatial pooling on the CLS token; whereas, convolutional networks undergo global spatial pooling at the very last

step, before the classifier. In this sense, one can isolate the pooling process from both kinds of networks and replace it by a new one. This raises the following questions:

1. *Can we derive a simple pooling process at the very last step of either convolutional or transformer encoders that improves over their default?*
2. *Can this process provide high-quality attention maps that delineate object boundaries, for both networks?*
3. *Do these properties hold under both supervised and self-supervised settings?*

To answer these questions, we develop a *generic pooling framework*, parametrized by: (a) the number of vectors in the pooled representation; (b) whether pooling is iterative or not; (c) mappings at every stage of the process; (d) pairwise similarities, attention function and normalization; and (e) a function determining the pooling operation.

We then formulate a number of existing pooling methods as instantiations of this framework, including (a) simple pooling mechanisms in convolutional networks [20, 48, 42, 40, 47], (b) iterative methods on more than one vectors like k-means [34, 33], (c) feature re-weighting mechanisms originally desinged as network components rather than pooling [23, 56], and (d) vision transformers [14, 49]. Finally, by discussing the properties of each group of methods, we derive a new, simple, attention-based pooling mechanism as a replacement of the default one for both convolutional and transformer encoders. SimPool provides high-quality attention maps that delineate object boundaries under both supervised and self-supervised settings, as shown for ViT-S [14] in Figure 1.

In summary, we make the following contributions:

1. We formulate a generic pooling framework that allows easy inspection and qualitative comparison of a wide range of methods.
2. We introduce a simple, attention-based, non-iterative, universal pooling mechanism that provides a single vector representation and answers all the above questions in the affirmative.
3. We conduct an extensive empirical study that validates the superior qualitative properties and quantitative performance of the proposed mechanism on standard benchmarks and downstream tasks.

2. Related Work

We discuss the most related work to pooling in convolutional networks and vision transformers. An extended version with more background is given in the appendix.

Convolutional networks Early convolutional networks [17, 27] are based on learnable *convolutional layers* interleaved with fixed *spatial pooling layers* that downsample. The same design remains until today [26, 46, 20, 32]. Apart from mapping to a new space, convolutional layers involve a form of local pooling and pooling layers commonly take average [27] or maximum [44, 26].

Early networks end in a fully-connected layer over a feature tensor of low resolution [27, 26, 46]. This evolved into spatial pooling, *e.g.* global / regional average followed by a classifier for category-level tasks like classification [29, 20] / detection [18], or global maximum followed by a pairwise loss [48] for instance-level tasks.

The spatial pooling operation at the end of the network is widely studied in instance level-tasks [3, 48, 42], giving rise to forms of *spatial attention* [24, 38, 6, 47, 36], In category-level tasks, it is more common to study *feature re-weighting* as components of the architecture [23, 56, 22]. The two are closely related because *e.g.* the weighted average is element-wise weighting followed by sum.

Pooling can be *spatial* [22, 38, 6, 47, 36], *over channels* [23], or both [24, 56]. CBAM [56] 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 not a pooling mechanism.

Vision transformers *Pairwise interactions* between features are forms of pooling or *self-attention* over the spatial [55, 4, 63, 41] or channel dimensions [8, 54]. Originating in language models [51], *vision transformers* [14] streamlined these approaches and dominated the architecture landscape. Several variants often bring back ideas from convolutional networks [31, 58, 19, 57, 15, 21, 61].

Transformers downsample only at the input, forming spatial *patch tokens*. Pooling is based on a learnable CLS token, which, beginning at the input space, undergoes the same self-attention operation with patch tokens and 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.

Few works that have studied beyond CLS for pooling are mostly limited to global average pooling (GAP) [31, 62, 50, 43]. CLS offers attention maps for free, however of low quality unless in a self-supervised setting [7], which is not well studied. Few works that attempt to rectify this in the supervised setting include a spatial entropy loss [39], shape distillation from convolutional networks [35] and skipping computation of self-attention [52].

We attempt to address these limitations and study pooling in convolutional networks, vision transformers, supervised and self-supervised alike. We derive a simple, attention-based, universal pooling mechanism, improving both performance and attention maps.

3. Method

We develop a generic pooling framework that encompasses many simple or more complex pooling methods, iterative or not, attention-based or not. We then examine a number of methods as instantiations of this framework. Finally, we discuss their properties and make particular choices in designing our solution.

3.1. A generic pooling framework

Preliminaries Let $\mathbf{X} \in \mathbb{R}^{d \times W \times H}$ be the 3-dimensional *feature tensor* obtained from the last layer of a network for a given input image, where d is the number of feature channels and W, H are the width and height. We represent the image by the *feature* matrix $X \in \mathbb{R}^{d \times p}$ by flattening the spatial dimensions of $\mathbf{X}$, where $p := W \times H$ is the number of spatial locations. Let $\mathbf{x}_i \in \mathbb{R}^p$ denote the i-th row of X, that is, corresponding to the 2-dimensional feature map in channel i, and $\mathbf{x}_{\bullet j} \in \mathbb{R}^d$ denote the j-th column of X, that is, the feature vector of spatial location j.

By $\mathbf{1}_n \in \mathbb{R}^n$, we denote the all-ones vector. Given an $m \times n$ matrix $A \geq 0$, by $\eta_1(A) := \mathrm{diag}(A\mathbf{1}_n)^{-1}A$ we denote row-wise ℓ_1-normalization; similarly, $\eta_2(A) := A\,\mathrm{diag}(\mathbf{1}_m^\top A)^{-1}$ for column-wise.

Pooling process The objective of pooling is to represent the image by one or more vectors, obtained by interaction with X, either in a single step or by an iterative process. We denote the pooling process by function $\pi : \mathbb{R}^{d \times p} \to \mathbb{R}^{d' \times k}$ and the output vectors by matrix $U = \pi(X) \in \mathbb{R}^{d' \times k}$, where d' is the number of dimensions, possibly $d' = d$, and k is the number of vectors. In the most common case of a single vector, $k = 1$, we denote U by $\mathbf{u} \in \mathbb{R}^{d'}$. We discuss here the general iterative process; single-step pooling is the special case where the number of iterations is 1.

Initialization We define $X^0 := X$ and make a particular choice for $U^0 \in \mathbb{R}^{d^0 \times k}$, where $d^0 := d$. The latter may depend on the input X, in which case it is itself a simple form of pooling or not; for example, it may be random or a learnable parameter over the entire training set.

Pairwise interaction Given U^t and X^t at iteration t, we define the *query* and *key* matrices

$$Q = \phi_Q^t(U^t) \in \mathbb{R}^{n^t \times k} \tag{1}$$

$$K = \phi_K^t(X^t) \in \mathbb{R}^{n^t \times p}. \tag{2}$$

Here, functions $\phi_Q^t : \mathbb{R}^{d^t \times k} \to \mathbb{R}^{n^t \times k}$ and $\phi_K^t : \mathbb{R}^{d^t \times p} \to \mathbb{R}^{n^t \times p}$ may be the identity, linear or non-linear mappings to a space of the same ($n^t = d^t$) or different dimensions. We let K, Q interact pairwise by defining the $p \times k$ matrix $S(K, Q) := ((s(\mathbf{k}_{\bullet i}, \mathbf{q}_{\bullet j}))_{i=1}^p)_{j=1}^k$, where $s : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ for any n is a similarity function. For example, s can be dot product, cosine similarity, or a decreasing function of some distance. In the case of dot product, $s(\mathbf{x}, \mathbf{y}) := \mathbf{x}^\top \mathbf{y}$ for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, it follows that $S(K, Q) = K^\top Q \in \mathbb{R}^{p \times k}$.

Attention We then define the *attention* matrix

$$A = h(S(K, Q)) \in \mathbb{R}^{p \times k}. \tag{3}$$

Here, $h : \mathbb{R}^{p \times k} \to [0, 1]^{p \times k}$ is a nonlinear function that may be elementwise, for instance relu or exp, normalization over rows or columns of $S(K, Q)$, or it may yield a form of correspondence or assignment between the columns of K and Q, possibly optimizing a cost function.

Attention-weighted pooling We define the *value* matrix

$$V = \phi_V^t(X^t) \in \mathbb{R}^{n^t \times p}. \tag{4}$$

Here, function $\phi_V^t : \mathbb{R}^{d^t \times p} \to \mathbb{R}^{n^t \times p}$ plays a similar role with ϕ_Q^t, ϕ_K^t. *Attention-weighted pooling* is defined by

$$Z = f^{-1}(f(V)A) \in \mathbb{R}^{n^t \times k}. \tag{5}$$

Here, $f : \mathbb{R} \to \mathbb{R}$ is a nonlinear elementwise function that determines the pooling operation, for instance, average or max-pooling. The product $f(V)A$ defines k linear combinations over the columns of $f(V)$, that is, the features at different spatial locations. If the columns of A are ℓ_1-normalized, then those are convex combinations. Thus, matrix A defines the weights of an averaging operation.

Output Finally, we define the output matrices corresponding to image features and pooling,

$$X^{t+1} = \phi_X^t(X^t) \in \mathbb{R}^{d^{t+1} \times p} \tag{6}$$

$$U^{t+1} = \phi_U^t(Z) \in \mathbb{R}^{d^{t+1} \times k}. \tag{7}$$

Functions $\phi_X^t : \mathbb{R}^{n^t \times p} \to \mathbb{R}^{d^{t+1} \times p}$ and $\phi_U^t : \mathbb{R}^{n^t \times k} \to \mathbb{R}^{d^{t+1} \times k}$ play a similar role with $\phi_Q^t, \phi_K^t, \phi_V^t$ but also determine the dimensionality d^{t+1} for the next iteration.

At this point, we may iterate by returning to the "pairwise interaction" step, or terminate, yielding U^{t+1} as U with $d' = d^{t+1}$. Non-iterative methods do not use ϕ_X^t.

3.2. A pooling landscape

Table 1 examines a number of pooling methods as instantiations of our framework. The objective is to get insight into their basic properties. How this table was obtained is detailed in the appendix.

Group 1 consists of simple methods with $k = 1$ that are not attention-based and have been studied in category-level tasks [20, 40] or mostly in instance-level tasks [48, 42, 47]. Here, the attention is a vector $\mathbf{a} \in \mathbb{R}^p$ and either is uniform or depends directly on X, by pooling over channels [47]. Most important is the choice of pooling operation by function f. Log-sum-exp [40] arises with $f(x) = e^{rx}$ with

#	METHOD	CAT	ITER	k	U^0	$\phi_Q(U)$	$\phi_K(X)$	$s(\mathbf{x},\mathbf{y})$	A	$\phi_V(X)$	$f(x)$	$\phi_X(X)$	$\phi_U(Z)$
1	GAP [20]	✓		1					$\mathbf{1}_p/p$	X	$f_{-1}(x)$		Z
	max [48]			1					$\mathbf{1}_p$	X	$f_{-\infty}(x)$		Z
	GeM [42]			1					$\mathbf{1}_p/p$	X	$f_\alpha(x)$		Z
	LSE [40]	✓		1					$\mathbf{1}_p/p$	X	e^{rx}		Z
	HOW [47]			1					$\mathrm{diag}(X^\top X)$	$\mathrm{FC}(\mathrm{avg}_3(X))$	$f_{-1}(x)$		$\eta^2(Z)$
2	OTK [34]	✓		k	U	U	X	$-\|\mathbf{x}-\mathbf{y}\|^2$	$\mathrm{SINKHORN}(e^{S/\epsilon})$	$\psi(X)$	$f_{-1}(x)$		Z
	k-means		✓	k	random	U	X	$-\|\mathbf{x}-\mathbf{y}\|^2$	$\eta_2(\arg\max_1(S))$	X	$f_{-1}(x)$	X	Z
	Slot [33]*	✓	✓	k	random	$W_Q U$	$W_K X$	$\mathbf{x}^\top\mathbf{y}$	$\eta_1(\boldsymbol{\sigma}_2(S/\sqrt{n}))$	$W_V X$	$f_{-1}(x)$	X	$\mathrm{MLP}(\mathrm{GRU}(Z))$
3	SE [23]	✓		1	$\pi_A(X)$	$\sigma(\mathrm{MLP}(U))$			$\mathbf{1}_p/p$	$\mathrm{diag}(\mathbf{q})X$		V	
	CBAM [56]*	✓		1	$\pi_A(X)$	$\sigma(\mathrm{MLP}(U))/d$	X	$\mathbf{x}^\top\mathbf{y}$	$\sigma(\mathrm{conv}_7(S))/p$	$\mathrm{diag}(\mathbf{q})X$		$V\,\mathrm{diag}(\mathbf{a})$	
4	ViT [14]*	✓	✓	1	U	$g_m(W_Q U)$	$g_m(W_K X)$	$\mathbf{x}^\top\mathbf{y}$	$\boldsymbol{\sigma}_2(S_i/\sqrt{d'})_{i=1}^m$	$g_m(W_V X)$	$f_{-1}(x)$	$\mathrm{MLP}(\mathrm{MSA}(X))$	$\mathrm{MLP}(W_U g_m^{-1}(Z))$
	CaiT [49]*	✓	✓	1	U	$g_m(W_Q U)$	$g_m(W_K X)$	$\mathbf{x}^\top\mathbf{y}$	$\boldsymbol{\sigma}_2(S_i/\sqrt{d'})_{i=1}^m$	$g_m(W_V X)$	$f_{-1}(x)$	X	$\mathrm{MLP}(W_U g_m^{-1}(Z))$
5	SimPool	✓		1	$\pi_A(X)$	$W_Q U$	$W_K X$	$\mathbf{x}^\top\mathbf{y}$	$\boldsymbol{\sigma}_2(S/\sqrt{d})$	$X-\min X$	$f_\alpha(x)$		Z

Table 1. A landscape of pooling methods. CAT: used in category-level tasks; ITER: iterative; *: simplified. π_A: GAP; σ: sigmoid; $\boldsymbol{\sigma}_2$: softmax over columns; η_2: column normalization; g_m: partitioning in m groups (see appendix). Cyan: ours; gray: common choices with ours; green: learnable; red: hyperparameter; blue: detailed in the appendix.

learnable scale r. For the rest, we define $f = f_\alpha$, where

$$f_\alpha(x) := \begin{cases} x^{\frac{1-\alpha}{2}}, & \text{if } \alpha \neq 1, \\ \ln x, & \text{if } \alpha = 1. \end{cases} \tag{8}$$

As studied by Amari [1], function f_α is defined for $x \geq 0$ ($\alpha \neq 1$) or $x > 0$ ($\alpha = 1$). It reduces to the maximum, quadratic mean (RMS), arithmetic mean, geometric mean, harmonic mean, and minimum for $\alpha = -\infty, -3, -1, 1, 3, +\infty$, respectively. It has been proposed as a transition from average to max-pooling [5] and is known as GeM [42], with $\gamma = (1-\alpha)/2 > 1$ being a learnable parameter.

Group 2 incorporates iterative methods with $k > 1$, including standard k-means, the soft-clustering variant Slot Attention [33] and optimal transport between U and X [34]. The latter is not formally iterative according to our framework, but the Sinkhorn algorithm is iterative internally.

Group 3 refers to methods introduced as modules within the architecture rather than pooling mechanisms [23, 56]. An interesting aspect is initialization of U^0 by *global average pooling* (GAP) on X:

$$\pi_A(X) := X\mathbf{1}_p/p = \frac{1}{p}\sum_{j=1}^{p} \mathbf{x}_{\bullet j} \in \mathbb{R}^d, \tag{9}$$

where $\mathbf{1}_p \in \mathbb{R}^p$ is the all-ones vector. Channel attention ($\phi_Q(U)$) and spatial attention (A) in CBAM [56] are based on a few layers followed by sigmoid, playing the role of a binary classifier (*e.g.* foreground/background); whereas, transformer-based attention uses directly the query and softmax normalization, respectively. Although not evident in the original formulation, we show in the appendix that there is pairwise interaction.

Group 4 refers to vision transformers [14, 49], which we reformulate in two separate streams, one for the CLS token, U, and another for the patch tokens, X. We observe that, what happens to the CLS token throughout the entire encoder, is an iterative pooling process. Moreover, although U is just one vector, multi-head attention splits it into m subvectors, where m is the number of heads. Thus, m is similar to k in k-means. The difference of CaiT [49] from ViT [14] is that this iteration happens only in the last couple of layers, with the patch embeddings X being fixed.

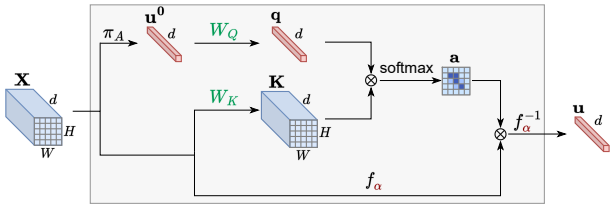

Figure 2. *Overview of SimPool.* Given an input tensor $\mathbf{X} \in \mathbb{R}^{d\times W\times H}$ flattened into $X \in \mathbb{R}^{d\times p}$ with $p := W \times H$ patches, one stream forms the initial representation $\mathbf{u}^0 = \pi_A(X) \in \mathbb{R}^d$ (12) by *global average pooling* (GAP), mapped by $W_Q \in \mathbb{R}^{d\times d}$ (13) to form the *query* vector $\mathbf{q} \in \mathbb{R}^d$. Another stream maps X by $W_K \in \mathbb{R}^{d\times d}$ (14) to form the *key* $K \in \mathbb{R}^{d\times p}$, shown as tensor $\mathbf{K}$. 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 X with $\mathbf{a}$ determining the weights and scalar function f_α determining the pooling operation (17).

3.3. SimPool

Group 5 of Table 1 is our method, SimPool. A schematic overview is given in Figure 2.

Pooling process We are striving for a simple design. While pooling into $k > 1$ vectors would yield a more discriminative representation, either these would have to be concatenated, as is the case of multi-head attention, or a particular similarity kernel would be needed beyond dot product, which we consider to be beyond the scope of this

work. We rather argue that it is the task of the encoder to learn a single vector representation of objects, even if those are composed of different parts. This argument is stronger when pre-training is performed on images mostly depicting one object, like ImageNet-1k.

We observe in Table 1 that only methods explicitly pooling into $k > 1$ vectors or implicitly using $m > 1$ heads are iterative. We explain why in the next paragraph. Following this insight, we perform pooling in a single step.

In summary, our solution is limited to a single vector $\mathbf{u} \in \mathbb{R}^d$ for pooling, that is, $k = 1$, and is non-iterative.

Initialization We observe in Table 1 that single-step attention-based methods in Group 3 initialize $\mathbf{u}^0$ by GAP. We hypothesize that, since attention is based on pairwise similarities, it is essential that $\mathbf{u}^0$ is chosen such that its similarities with X are maximized on average, which would help to better discriminate between foreground (high similarity) and background (low similarity). Indeed, for $s(\mathbf{x}, \mathbf{y}) = -\|\mathbf{x} - \mathbf{y}\|^2$, the sum of squared Euclidean distances of each column $\mathbf{x}_{\bullet i}$ of X to $\mathbf{u} \in \mathbb{R}^d$

$$J(\mathbf{u}) = \frac{1}{2} \sum_{i=1}^{p} \|\mathbf{x}_{\bullet i} - \mathbf{u}\|^2 \tag{10}$$

is a convex distortion measure with unique minimum the average of vectors $\{\mathbf{x}_{\bullet i}\}$

$$\mathbf{u}^* := \arg\min_{\mathbf{u} \in \mathbb{R}^d} J(\mathbf{u}) = \frac{1}{p} \sum_{i=1}^{p} \mathbf{x}_{\bullet i} = \pi_A(X), \tag{11}$$

which can be found in closed form. By contrast, for $k > 1$ vectors, distortion can only be minimized iteratively, *e.g.* by k-means. We therefore choose:

$$\mathbf{u}^0 = \pi_A(X) = X\mathbf{1}_p/p. \tag{12}$$

Pairwise interaction, attention We follow the attention mechanism of transformers, in its simplest possible form. In particular, we use a single head, $m = 1$, like Slot Attention [33] (which however uses k vectors). We find that the query and key mappings are essential in learning where to attend as a separate task from learning the representation for the given task at hand. In particular, we use linear mappings ϕ_Q, ϕ_K with learnable parameters $W_Q, W_K \in \mathbb{R}^{d \times d}$ respectively:

$$\mathbf{q} = \phi_Q(\mathbf{u}^0) = W_Q \mathbf{u}^0 \in \mathbb{R}^d \tag{13}$$

$$K = \phi_K(X) = W_K X \in \mathbb{R}^{d \times p}. \tag{14}$$

As in transformers, we define pairwise similarities as dot product, that is, $S(K, \mathbf{q}) = K^\top \mathbf{q} \in \mathbb{R}^{p \times k}$, and attention as scaled softmax over columns (spatial locations), that is, $h(S) := \boldsymbol{\sigma}_2(S/\sqrt{d})$:

$$\mathbf{a} = \boldsymbol{\sigma}_2\left(K^\top \mathbf{q}/\sqrt{d}\right) \in \mathbb{R}^p, \tag{15}$$

where $\boldsymbol{\sigma}_2(S) := \eta_2(\exp(S))$ and exp is taken elementwise.

Attention-weighted pooling As shown in Table 1, the average pooling operation ($f = f_{-1}$) is by far the most common. However, the more general function f_α (8) has shown improved performance in instance-level tasks [42]. For $\alpha < -1$ ($\gamma > 1$) in particular, it yields an intermediate operation between average and max-pooling. The latter is clearly beneficial when feature maps are sparse, because it better preserves the non-zero elements.

We adopt $f = f_\alpha$ for its genericity: the only operation that is not included as a special case in Table 1 is log-sum-exp [40]. This choice assumes $X \geq 0$. This is common in networks ending in relu, like ResNet [20], which is also what makes feature maps sparse. However, vision transformers and modern convolutional networks like ConvNeXt [32] do not end in relu; hence X has negative elements and is not necessarily sparse. We therefore define

$$V = \phi_V(X) = X - \min X \in \mathbb{R}^{d \times p}, \tag{16}$$

where the minimum is taken over all elements of X, such that f_α operates only on non-negative numbers.

We also define $\mathbf{u} = \phi_U(\mathbf{z}) = \mathbf{z}$ and the output dimension is $d' = d$. Thus, the mappings ϕ_V, ϕ_U are parameter-free. The argument is that, for average pooling for example ($f = f_{-1}$ in (5)), any linear layers before or after pooling would commute with pooling, thus they would form part of the encoder rather than the pooling process. Moreover, Table 1 shows that ϕ_U is non-identity only for iterative methods.

In summary, we define SimPool (SP) as

$$\mathbf{u} = \pi_{\text{SP}}(X) := f_\alpha^{-1}(f_\alpha(V)\mathbf{a}) \in \mathbb{R}^d, \tag{17}$$

where $V \in \mathbb{R}^{d \times p}$ is the value (16) and $\mathbf{a} \in \mathbb{R}^p$ is the attention map (15). Parameter α is learned in GeM [42], but we find that treating it as a hyperparameter better controls the quality of the attention maps.

4. Experiments

4.1. Datasets, networks and evaluation protocols

Supervised pre-training We train ResNet-18, ResNet-50 [20], ConvNeXt-S [32], ViT-S and ViT-B [14] for *image classification* on ImageNet-1k. For the analysis subsection 4.2 and ablation subsection 4.4, we train ResNet-18 on the first 20% of training examples per class of ImageNet-1k [11] (called ImageNet-20%) for 100 epochs. For the benchmark of subsection 4.3, 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, all on the 100% of ImageNet-1k. We evaluate on the full validation set in all cases and measure top-1 classification accuracy. The baseline is the default per network, *i.e.* GAP for convolutional networks and CLS token for transformers.

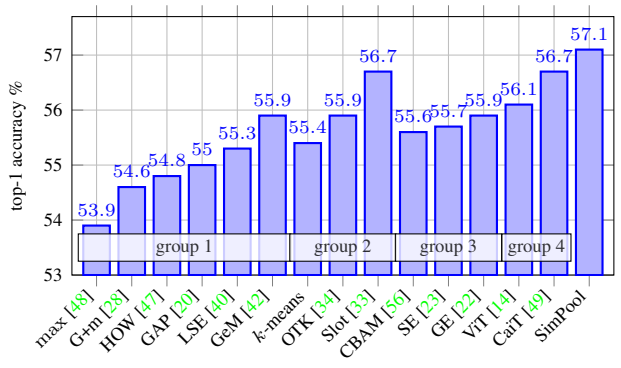

Figure 3. *Image classification* on ImageNet-20. Supervised training of ResNet-18 for 100 epochs.

METHOD	EP	RESNET-50	CONVNEXT-S	VIT-S	VIT-B
Baseline	100	77.4	81.1	72.7	74.1
CaiT [49]	100	77.3	81.2	72.6	-
Slot [33]	100	77.3	80.9	72.9	-
GE [22]	100	77.6	81.3	72.6	-
SimPool	100	**78.0**	**81.7**	**74.3**	**75.1**
Baseline	300	78.1†	83.1	77.9	-
SimPool	300	**78.7**†	**83.5**	**78.7**	-

Table 2. *Image classification* top-1 accuracy (%) on ImageNet-1k. Supervised pre-training for 100 and 300 epochs. Best competitors selected per group from Figure 3. Baseline: GAP for convolutional, CLS for transformers; EP: epochs; †: 200 epochs.

Self-supervised pre-training On the 100% of ImageNet-1k, we train DINO [7] with ResNet-50, ConvNeXt-S and ViT-S for 100 epochs. We evaluate on the validation set by k-NN and *linear probing* on the training set. For *linear probing*, we train a linear classifier on top of features as in DINO [7]. For k-NN [59], we freeze the model and extract features, then use a k-nearest neighbor classifier with $k = 10$.

Downstream tasks We fine-tune supervised and self-supervised ViT-S on CIFAR-10 [25], CIFAR-100 [25] and Oxford Flowers [37] for *image classification*, measuring top-1 classification accuracy. We perform *object localization* without fine-tuning using supervised and self-supervised ViT-S on CUB [53] and ImageNet-1k, measuring MaxBoxAccV2 [9]. We perform *object discovery* without fine-tuning using self-supervised ViT-S with DINOSEG [7] and LOST [45] on VOC07 [16], VOC12 [16] and COCO [30], measuring CorLoc [12]. We validate *robustness* against background changes using ViT-S on ImageNet-9 [60] and its variations. We use the linear head and linear probe for supervised and self-supervised ViT-S, respectively, measuring top-1 classification accuracy.

In the appendix, we provide implementation details, more benchmarks, ablations and visualizations.

4.2. Experimental Analysis

Figure 3 evaluates different methods in groups following Table 1, regardless of their original design for (a) pooling or not, (b) different tasks, *e.g.* instance-level or category-level, (c) different networks, *e.g.* convolutional or transformers.

Group 1 consists of simple pooling methods with: (a) no parameters: GAP [29], max [48], GAP+max [28]; and (b) scalar parameter: GeM [42] and LSE [40]. HOW [47] is the only method to use (parameter-free) attention. GeM is performing the best, with LSE following second. These methods are inferior to those in other groups.

Group 2 incorporates methods with $k > 1$ vectors. We set $k = 3$ and take the maximum of the 3 logits per class. OTK and Slot use attention. Slot attention [33] works best, outperforming k-means by 1.3%.

Group 3 refers to parametric attention-based methods, weighting features based on their importance for the task: CBAM [56], Squeeze-Excitation [23] and Gather-Excite [22]. While originally designed as components within the architecture, we adapt them to pooling by GAP at the end. Gather-Excite [22] performs best.

Group 4 refers to parametric attention-based methods found in vision transformers. ViT [14] refers to multi-head self-attention learnable CLS and four heads, which we incorporate as a single layer at the end of the model. CaiT [49] is the same but using only cross-attention between CLS and patch embeddings. CaiT performs the best.

SimPool outperforms all other methods. Seeing this experiment as a tournament, we select the best performing method of each group and qualify it for the benchmark of subsection 4.3.

4.3. Benchmark

Image Classification Table 2 compares SimPool with baseline and tournament winners per group of subsection 4.2 on supervised pre-training for classification. For 100 epochs, SimPool outperforms all methods, consistently improving the baseline by 0.6% using convolutional networks, 1.6% using ViT-S and 1.0% using ViT-B. Gather-Excite [22] improves over the baseline only on convolutional networks, while Slot [33] only on ViT-S. CaiT improves over the baseline only for ConvNeXt-S. By contrast, SimPool improves everywhere. For more than 100 epochs, SimPool improves the baseline by 0.5% using ResNet-50, 0.4% using ConvNeXt-S and 0.8% using ViT-S.

Table 3 evaluates self-supervised pre-training for 100 epochs. SimPool improves over the baseline by 2.0% k-NN and 1.4% linear probing on ResNet-50; 3.7% k-NN and 4.0% linear probing on ConvNeXt-S; and 0.9% k-NN and 1.3% linear probing on ViT-S.

METHOD	EP	RESNET-50		CONVNEXT-S		VIT-S	
		k-NN	PROB	k-NN	PROB	k-NN	PROB
Baseline	100	61.8	63.0	65.1	68.2	68.9	71.5
SimPool	100	**63.8**	**64.4**	**68.8**	**72.2**	**69.8**	**72.8**

Table 3. *Image classification* top-1 accuracy (%) on ImageNet-1k. Self-supervised pre-training with DINO [7] for 100 epochs. Baseline: GAP for convolutional, CLS for transformers.

METHOD	SUPERVISED			SELF-SUPERVISED		
	CF-10	CF-100	FL	CF-10	CF-100	FL
Baseline	98.1	86.0	97.1	98.7	89.8	98.3
SimPool	**98.4**	**86.2**	**97.4**	**98.9**	**89.9**	**98.4**

Table 4. *Image classification* accuracy (%) with fine-tuning for classification for 1000 epochs. ViT-S pre-trained on ImageNet-1k for 100 epochs. Self-supervision with DINO [7]. CF-10: CIFAR-10 [25], CF-100: CIFAR-100 [25], FL: Flowers[37].

METHOD	SUPERVISED		SELF-SUPERVISED	
	CUB	IMAGENET	CUB	IMAGENET
Baseline	63.1	53.6	82.7	62.0
SimPool	**77.9**	**64.4**	**86.1**	**66.1**
Baseline@20	62.4	50.5	65.5	52.5
SimPool@20	**74.0**	**62.6**	**72.5**	**58.7**

Table 5. *Localization accuracy* MaxBoxAccV2 on CUB test and ImageNet-1k validation set. ViT-S pre-trained on ImageNet-1k for 100 epochs. Self-supervision with DINO [7]. @20: at epoch 20.

Fine-tuning for classification Table 4 evaluates fine-tuning for classification on different datasets of a supervised and a self-supervised ViT-S. SimPool brings a small improvement over the baseline in all cases.

Object localization Accurate localization can have a significant impact on classification accuracy, particularly under multiple objects, complex scenes and background clutter. Table 5 evaluates localization accuracy under both supervision settings. SimPool significantly improves the baseline by up to 7% MaxBoxAccV2 when self-supervised and up to 14% when supervised. In the latter case, the gain is already up to 12% at epoch 20.

Unsupervised object discovery Table 6 studies LOST [45], which uses the raw features of a vision transformer pre-trained using DINO [7] for unsupervised single-object discovery, as well as the baseline DINO-seg [45, 7], which uses the attention maps instead. SimPool significantly outperforms the baseline on all datasets by up to 25.2% CorLoc for DINO-seg and 5.6% for LOST on VOC12. Again, the gain is significant already at the first 20 epochs.

Background changes We evaluate robustness to the background changes using IN-9 [60] dataset. Table 7 shows

METHOD	DINO-SEG [45, 7]			LOST [45]		
	VOC07	VOC12	COCO	VOC07	VOC12	COCO
Baseline	30.8	31.0	36.7	55.5	59.4	46.6
SimPool	**53.2**	**56.2**	**43.4**	**59.8**	**65.0**	**49.4**
Baseline@20	14.9	14.8	19.9	50.7	56.6	40.9
SimPool@20	**49.2**	**54.8**	**37.9**	**53.9**	**58.8**	**46.1**

Table 6. *Object discovery* CorLoc. ViT-S pre-trained on ImageNet-1k for 100 epochs. Self-supervision with DINO [7]. @20: at epoch 20.

METHOD	OF	MS	MR	MN	NF	OBB	OBT	IN-9
SUPERVISED								
Baseline	66.4	79.1	67.4	65.5	37.2	12.9	15.2	92.0
SimPool	**71.8**	**80.2**	**69.3**	**67.3**	**42.8**	**15.2**	**15.6**	**92.9**
SELF-SUPERVISED + LINEAR PROBING								
Baseline	**87.3**	87.9	78.5	76.7	47.9	**20.0**	**16.9**	95.3
SimPool	**87.3**	**88.1**	**80.6**	**78.7**	**48.2**	17.8	16.7	**95.6**

Table 7. *Background robustness* on IN-9 [60] and its variations; more details in the appendix. ViT-S pre-trained on ImageNet-1k for 100 epochs. Self-supervision with DINO [7].

that SimPool improves over the baseline under both supervision settings with only 2 out of 8 exceptions under DINO [7] pre-training. The latter is justified, given that none of the foreground objects or masks are present in these settings.

	RESNET-18		RESNET-50		CONVNEXT-S		VIT-S	
	#PAR	FLO	#PAR	FLO	#PAR	FLO	#PAR	FLO
Baseline	11.7	1.82	25.6	4.13	50.2	8.68	22.1	4.24
CaiT	18.0	1.85	75.9	4.60	57.3	8.75	23.8	4.29
Slot	14.6	1.87	71.7	4.89	56.7	8.79	23.7	4.30
GE	11.7	1.83	26.1	4.15	50.3	8.69	22.1	4.25
SimPool	12.2	1.84	33.9	4.34	51.4	8.71	22.3	4.26

Table 8. *Computation resources* on Imagenet-1k, with $d = 512$ (ResNet-18), 2048 (ResNet-50), 768 (ConvNeXt), 384 (ViT-S). #PAR: number of parameters, in millions; FLO: GFLOPS.

Computation resources Table 8 shows the number of parameters and floating point operations per second for the best competitors of Figure 3. Resources depend on the embedding dimension d. SimPool is higher than the baseline but not the highest.

Performance *vs.* parameters Table 9 aims to answer the question of how much the performance improvement of SimPool is due to parameters of the query and key mappings. Interestingly, ViT-S works better with GAP than the default CLS. SimPool adds 0.2M parameters to the network. For fair comparison, we remove blocks from the network

NETWORK	POOLING	DEPTH	INIT	ACCURACY	#PARAMS
BASE	GAP	12	12	73.3	22.1M
BASE	CLS	12	0	72.7	22.1M
BASE + 1		13	0	73.2	23.8M
BASE + 2		14	0	73.7	25.6M
BASE + 3		15	0	73.8	27.4M
BASE + 4		16	0	73.9	29.2M
BASE + 5		17	0	**74.6**	30.9M
BASE	SimPool	12	12	**74.3**	22.3M
BASE − 1		11	11	73.9	20.6M
BASE − 2		10	10	73.6	18.7M
BASE − 3		9	9	72.5	17.0M

Table 9. *Trade-off between performance and parameters*. Supervised pre-training of ViT-S on ImageNet-1k for 100 epochs. INIT: Initial layer of pooling token. BASE: original network. BASE+b (BASE−b): b blocks added to (removed from) the network.

(BASE) when using SimPool and add blocks when using CLS. We find that, to exceed the accuracy of BASE SimPool, BASE CLS needs 5 extra blocks, *i.e.*, 9M more parameters. Equally interestingly, removing 3 blocks from BASE SimPool is only slightly worse than BASE CLS, having 5M fewer parameters.

4.4. Ablation study

We ablate the design and components of SimPool. More ablations are found in the appendix. In particular, for function f_α (8), we set $\gamma = 2$ for convolutional networks and $\gamma = 1.25$ for transformers by default, where $\gamma = (1-\alpha)/2$ is a hyperparameter.

Design In Table 10 (left), we ablate (a) the attention function h (3); (b) the number of iterations with shared parameters at every iteration (LAYERS) or not (ITER); (c) the initialization U^0; (d) the pairwise similarity function s; (e) the number k of pooled vectors, obtained by k-means instead of GAP. We also consider queries and keys sharing the same mapping, $W_Q = W_K$. We observe that multi-head, few iterations and initialization by $\mathrm{diag}(X^\top X)$ perform slightly worse, without adding any extra parameters, while setting $W_Q = W_K$ performs slightly worse, having 50% less parameters.

Linear and LayerNorm layers In Table 10 (right), we systematically ablate linear and LayerNorm (LN) [2] layers on query q, key k and value v. We strive for performance and quality while at the same time having a small number of components and parameters. In this sense, we choose the setup that includes linear layers on q, k and LN on k, v, yielding 56.6 accuracy. We observe that having linear and LN layers everywhere performs best under classification accuracy. However, this setup has attention maps of lower quality and more parameters.

ABLATION	OPTIONS	ACC	LINEAR	LN	ACC
			Q K V	Q K V	
$h(S)$	$\boldsymbol{\sigma}_2(S_i/\sqrt{d})_{i=1}^m$	56.6			
	$\eta_2(\boldsymbol{\sigma}_1(S/\sqrt{d}))$	55.6	✓ ✓ ✓	✓ ✓ ✓	**57.0**
LAYERS	3	56.8	✓ ✓	✓ ✓ ✓	**56.6**
	5	55.9	✓	✓ ✓ ✓	56.5
ITER	3	56.5	✓	✓ ✓ ✓	56.4
	5	56.4		✓ ✓ ✓	55.6
U^0	U	56.3	✓ ✓	✓ ✓	56.3
	$\mathrm{diag}(X^\top X)$	56.6	✓ ✓	✓	56.0
$s(\mathbf{x}, \mathbf{y})$	$-\|\mathbf{x}-\mathbf{y}\|^2$	56.5	✓ ✓	✓	56.2
	cosine	56.3	✓ ✓	✓ ✓	**56.6**
k (max)	2	56.5	✓ ✓	✓	56.4
	5	56.4	✓ ✓	✓ ✓	56.2
k (concat)	2	56.5		✓ ✓	56.2
	5	55.9	✓ ✓		54.4
ϕ_Q, ϕ_K	$W_Q = W_K$	56.4			54.5
SimPool		**57.1**	GAP		55.0

Table 10. SimPool ablation on ImageNet-20% using ResNet-18 trained for 100 epochs. Ablation of (left) design; (right) linear and LayerNorm (LN) [2] layers. q, k, v: query, key, value. $\boldsymbol{\sigma}_2(S_i/\sqrt{d})_{i=1}^m$: same as our default, but with multi-head attenion, $m = 4$ heads; k (max): maximum taken over output logits; k (concat): concatenation and projection to the same output dimensions d'. Green: learnable parameter; blue: winning choice per group of experiments; Cyan: Our chosen default. Using pooling operation $f = f_\alpha$ (8) (left); $f = f_{-1}$ (right).

5. Conclusion

We have introduced SimPool, a simple, attention-based pooling mechanism that acts at the very last step of either convolutional or transformer encoders, delivering highly superior quantitative results on several benchmarks and downstream tasks. In addition, SimPool delivers decent attention maps in both convolutional and transformer networks under both supervision and self-supervision with remarkable improvement in delineating object boundaries for supervised transformers. Despite this progress, we believe that investigating why the standard CLS-based attention fails under supervision deserves further study.

Acknowledgements This work was supported by the Hellenic Foundation for Research and Innovation (HFRI) under the BiCUBES project (grant: 03943). It was also partially supported by the RAMONES and iToBos EU Horizon 2020 projects, under grants 101017808 and 965221, respectively. NTUA thanks NVIDIA for the support with the donation of GPU hardware.

References

[1] Shun-ichi Amari. Integration of stochastic models by minimizing α-divergence. *Neural computation*, 19(10):2780–2796, 2007. 4

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 8

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

[4] 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

[5] 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. 4

[6] 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. 2

[7] 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. 1, 2, 6, 7

[8] 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

[9] 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. 6

[10] 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

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. Ieee, 2009. 1, 5

[12] Thomas Deselaers, Bogdan Alexe, and Vittorio Ferrari. Localizing objects while learning their appearance. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*, pages 452–466. Springer, 2010. 6

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1

[14] 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. 1, 2, 4, 5, 6

[15] 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

[16] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–308, 2009. 6

[17] 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, 2

[18] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision*, 2015. 2

[19] 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

[20] 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, 2, 3, 4, 5, 6

[21] 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

[22] 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. 2, 6

[23] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2, 4, 6

[24] Yannis Kalantidis, Clayton Mellina, and Simon Osindero. Cross-dimensional weighting for aggregated deep convolutional features. In *European Conference on Computer Vision Workshops*, 2016. 2

[25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6, 7

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*. 2012. 2

[27] 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. 2

[28] Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks:

Mixed, gated, and tree. In *Artificial intelligence and statistics*, pages 464–472. PMLR, 2016. 6

[29] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013. 2, 6

[30] 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. 6

[31] 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

[32] 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, 2, 5

[33] 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. 2, 4, 5, 6

[34] 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, 4, 6

[35] 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

[36] 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. 2

[37] 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. 6, 7

[38] 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. 2

[39] 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

[40] 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. 2, 3, 4, 5, 6

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

[42] 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, 2, 3, 4, 5, 6

[43] 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

[44] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *Computer Vision and Pattern Recognition*, 2005. 2

[45] 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. 6, 7

[46] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 2

[47] 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. 2, 3, 4, 6

[48] 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. 2, 3, 4, 6

[49] 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, 4, 6

[50] 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

[51] 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

[52] 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

[53] 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. 6

[54] 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

[55] 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

[56] 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. 2, 4, 6

[57] 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

[58] 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

[59] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018. 6

[60] 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. 6, 7

[61] 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

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

[63] 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