Kernelized Few-shot Object Detection with Efficient Integral Aggregation

Shan Zhang[*,†], Lei Wang[♦], Naila Murray[♣], Piotr Koniusz[*,§,†]
[†]Australian National University [♦]University of Wollongong [§]Data61/CSIRO [♣]Meta AI Research

[†]firstname.lastname@anu.edu.au, [♦]leiw@uow.edu.au, [♣]murrayn@fb.com

Abstract

*We design a Kernelized Few-shot Object Detector by leveraging kernelized matrices computed over multiple proposal regions, which yield expressive non-linear representations whose model complexity is learned on the fly. Our pipeline contains several modules. An Encoding Network encodes support and query images. Our Kernelized Autocorrelation unit forms the linear, polynomial and RBF kernelized representations from features extracted within support regions of support images. These features are then cross-correlated against features of a query image to obtain attention weights, and generate query proposal regions via an Attention Region Proposal Net. As the query proposal regions are many, each described by the linear, polynomial and RBF kernelized matrices, their formation is costly but that cost is reduced by our proposed Integral Region-of-Interest Aggregation unit. Finally, the Multi-head Relation Net combines all kernelized (second-order) representations with the first-order feature maps to learn support-query class relations and locations. We outperform the state of the art on novel classes by **3.8**%, **5.4**% and **5.7**% mAP on PASCAL VOC 2007, FSOD, and COCO.*

1. Introduction

CNN object detectors [8, 29–31] require thousands of manually annotated images for training. Their performance drops during adaptation to novel classes if samples are few.

In contrast, Few-shot Learning (FSL) methods rapidly adapt to new visual concepts [39, 41, 43] but off-the-shelf FSL methods perform classification rather than Few-shot Object Detection (FSOD). As queries in FSOD contain multiple objects of various categories and FSOD detectors have to predict class labels and locations of objects in a query image, effective techniques capturing query-support similarities across multiple Regions-of-Interest (RoI) are required.

FSOD models [2, 6, 11, 12, 52, 58] are trained with so-called training episodes containing samples of common objects (*i.e*. base classes). Testing episodes contain support images of rare objects (*i.e*. novel classes) and query images in which these rare objects must be recognized/localized. Fan *et al*. [6] introduced into FSOD a Region Proposal Network (RPN), termed Attention RPN (ARPN)[1]. ARPN cross-correlates average-pooled features from support regions with features of the query image, which produces an attention map over the feature tensor of query image. However, average pooling (a first-order statistic) retains less information compared to higher-order statistics. PNSD [58] improves [6] by second-order pooling but is limited to so-called linear correlations (autocorrelation matrix). To address this limitation, we use kernelized covariance matrices [57] and Reproducing Kernel Hilbert Space (RKHS) kernels [38] which capture non-linear patterns. Kernels induce regularization *e.g*., an RBF kernel with a small (resp. large) radius captures a complex (resp. simple) decision boundary. However, as generating kernel matrices is computationally expensive, they are rarely used in detection.

We propose a novel feature representation which leverages the expressiveness and regularization capabilities of kernels, while enjoying an efficient implementation. The key to this efficiency is a novel Integral Region-of-Interest Aggregation (IRA) scheme for fast kernelization. We further accelerate IRA by count sketching [47], an unsupervised dimensionality reduction technique with a favourable property of implicitly performing feature augmentations. As the variance introduced by sketching is inversely-proportional to its size, it boosts the accuracy and computational speed, as described in Section 4. Our pipeline is shown in Figure 1. Our contributions are listed below:

i. We propose two types of kernelized representations used conjointly for FSOD that capture non-linear correlation patterns, obtained from candidate regions by computationally efficient Integral Region-of-Interest Aggregation (IRA). The performance and speed of IRA are boosted with count sketching and its inverse to facilitate the practical use of kernelization in FSOD (generating hundreds of kernels per image).

*Equal contribution. PK is the corresponding author.
Code: https://github.com/ZS123-lang/KFSOD.

[1]PSND [58] calls this module Hyper Attention RPN (HARPN) but FSOD-ARPN [6] calls it ARPN. We adopt the ARPN name for brevity.

ii. We equip our network with MLP units which learn the kernel hyper-parameters on the fly to adjust the learning complexity of kernelized representations to the data. We partially whiten matrices using Spectral Power Normalization [16] whose hyper-parameters are learnt via another MLP to extract the most informative features that concentrate along diagonals of kernel matrices.

iii. We redesign a Multi-head Relation Network to combine the first-order spatially-ordered features of support and query regions with the spatially orderless kernelized representations that contain higher-order statistics.

Advantages of RKHS kernelization in FSOD. We note that (i) kernels are very good at capturing non-linear relationships between feature channels of each candidate bounding box, (ii) kernels factor out spatial order while keeping rich statistics about each region, thus matching similar objects that vary in physical location, orientation, viewpoint is easy due to the shift-invariance, (iii) kernels let control the model complexity w.r.t. the region size and visual complexity, (iv) typical FSOD head uses either shift-variant or average pooled representations (we combine both).

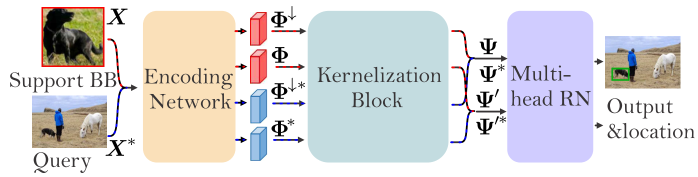


Figure 1. Kernelized Few-Shot Object Detector (KFSOD): We pass ground truth support bounding boxes $\mathbf{X}$ and query image $\mathbf{X}^*$ to Encoding Network (Fig. 2a). Feature Maps ($\mathbf{\Phi}^{\downarrow}$ and $\mathbf{\Phi}$ are of low and high resolution) are passed to the Kernelization Block (Fig. 3a), with the Attention Region Proposal Network (ARPN) to produce query region proposals, the Kernelized Autocorrelation (KA) and the Integral RoI Aggregation (IRA) units which accelerate kernelization. First-order and kernelized representations $\mathbf{\Psi}$ and $\mathbf{\Psi}'$ are fed to Multi-head Relation Network (MRN) in Fig. 2b.

2. Related Works

Below, we describe popular object detection and FSL algorithms and prerequisites such as second-order pooling.

Object Detection. One-stage detectors perform a regression to bounding box annotations [25, 29, 30]. Two-stage detectors, *e.g*., by R-CNN [31], generate class-agnostic region proposals which are then classified by a classification head [8, 31]. SNIPER [37] uses multi-scale training. DETR [1] and its variants [3, 59, 61] are anchor-free pipelines. Object detectors use large-scale datasets and fixed classes.

Few-shot Learning. Metric-learning FSL [13, 13, 27, 33, 34, 41] learns image-to-image similarity to generalize to novel classes. Prototypical Networks [39] compute distances between a datapoint and prototypes of each class. MAML [7] performs meta-learning. Others use subspaces [35, 60], gradient modulation [36] and self-supervision [55, 56].

Few-shot Object Detection. In [11], a single-stage FSOD detector reweights base model features to adapt to new classes. Meta R-CNN [52], a two-stage detector, reweights RoI features in the detection head. Based on a balanced dataset, TFA [45] fine-tunes a two-stage model. MPSR [49] improves TFA by training over multiple scales of positive samples. NP-RepMet [53] uses negative- and positive-representative learning via triplet losses that bootstrap the classifier. FSOD-ARPN [6] proposed a FSOD network with attention and a multi-relation head to score pairwise object similarity. PNSD [58], inspired by FSOD-ARPN [6], uses second-order representations to describe proposal regions.

Multi-path and Feature Groups. ResNeXt [50] adopts a group convolution [20] in the ResNet bottleneck block. SK-Net [22], based on SE-Net [9], uses the feature-map attention across two network branches. Bilinear pooling [32] correlates features extracted from two regions, whereas ReDRO [28] samples groups of features (akin to dropout) to apply the matrix square root over resulting submatrices.

Second-order Pooling (SOP). Texture recognition [51], Region Covariance Descriptors [42], and object classification [16] employ SOP. Fine-grained classification [19] and SoSN FSL [54] use SOP with Power Normalization [16].

Power Normalization (PN). Burstiness, "the property that a given visual element appears more times in an image than a statistically independent model would predict" [10]. Burstiness can be limited by PN [17] a feature detector using the cumulative distribution function of a binomial distribution to factor out feature counts [16, 17, 19]. The Fast Spectral MaxExp operator, MaxExp(F) [18], and Tensor Power-Euclidean (TPE) metric [15] reverse the heat diffusion of the implicit loopy graph of the second-order matrix to a desired past state [38]. In this work we decorrelate kernelized matrices to a desired level using MaxExp(F)[2], and extract the diagonals of matrices to use as representations.

3. Prerequisites

Notations. Let $\mathbf{x} \in \mathbb{R}^d$ be a d-dimensional feature vector. I_N stands for the index set $\{1, 2, \cdots, N\}$. We define $\mathbf{1} = [1, ..., 1]^T$ (*i.e*. the 'all ones' vector). Capitalised bold symbols such as $\mathbf{\Phi}$ denote matrices, lowercase bold symbols such as $\boldsymbol{\phi}$ denote vectors, and regular fonts denote scalars *e.g*., $\Phi_{i,j}$, ϕ_i, n or Z. $\Phi_{i,j}$ is the (i, j)-th entry of $\mathbf{\Phi}$. Diag$(\cdot)$ puts the matrix diagonal into a vector, Diag$^\dagger(\cdot)$ embeds a vector to form a diagonal matrix, $[x_\iota]_{\iota \in \mathcal{I}_N}$ stacks $x_1, \cdots, x_N$ into a vector, $\delta(x) = 1$ if $x = 0$, $\delta(x) = 0$ if $x \neq 0$, and $\mathbb{I}$ is the identity matrix.

(Eigenvalue) Power Normalization ((E)PN). MaxExp(F), a state-of-the-art EPN [18], is defined as

$$g(\lambda; \eta) = 1 - (1 - \lambda)^\eta \tag{1}$$

[2]Please note this is a prerequisite tool we use, not a contribution per se.

on the ℓ_1-norm normalized spectrum from SVD ($\lambda_i := \lambda_i/(\sum_{i'} \lambda_{i'}+\varepsilon)$), and on symmetric positive semi-definite matrices (PSD) as

$$\widehat{\mathcal{G}}_{\text{MaxExp}}(\mathbf{K};\eta) = \mathbb{I}-(\mathbb{I}-\mathbf{K})^{\eta}\,. \tag{2}$$

Here, $\mathbf{K}$ is a trace-normalized SPD matrix, $\varepsilon \geq 0$ is a small constant, and $\eta \geq 1$ is used to adjust the degree to which features are decorrelated. Larger values of η result in greater decorrelation. As a result, rarer and more unique visual features are less overshadowed by large areas of visually-repetitive stimuli. $\widehat{\mathcal{G}}_{\text{MaxExp}}$ is followed by the element-wise PN, called SigmE [18]:

$$\mathcal{G}_{\text{SigmE}}(p;\eta') = 2/(1+e^{-\eta' p})-1, \tag{3}$$

where p takes each output entry of Eq. (2), $\eta' \geq 1$ controls detecting feature occurrence *vs*. feature counting trade-off.

Count Sketches. Count sketching [47] is an unsupervised dimensionality reduction technique which comes handy in reducing the size of our kernelized representations, described in Section 4. Let K and K' be the sizes of the input and sketched output. Let vector $\mathbf{h} \in \mathcal{I}^d_{K'}$ contain K uniformly drawn integer numbers from $\{1, \cdots, K'\}$ and vector $\mathbf{s} \in \{-1, 1\}^K$ contain K uniformly drawn values from $\{-1, 1\}$. The sketch projection matrix $\mathbf{P} \in \{-1, 0, 1\}^{K' \times K}$ is given as $P_{ij}(\boldsymbol{h}, \boldsymbol{s}) = s_j \cdot \delta(h_j - i)$ and the sketch projection Proj : $\mathbb{R}^K \to \mathbb{R}^{K'}$ is a linear operation $\text{Proj}_{\boldsymbol{h},\boldsymbol{s}}(\boldsymbol{\phi}) = \mathbf{P}(\boldsymbol{h}, \boldsymbol{s})\boldsymbol{\phi}$ (or $\text{Proj}(\boldsymbol{\phi}) = \mathbf{P}\boldsymbol{\phi}$). Weinberger *et al.* [47] showed that count sketches are unbiased estimators of the inner product $\mathbb{E}_{\boldsymbol{h},\boldsymbol{s}}(\langle \text{Proj}_{\boldsymbol{h},\boldsymbol{s}}(\boldsymbol{\phi}), \text{Proj}_{\boldsymbol{h},\boldsymbol{s}}(\boldsymbol{\phi}')\rangle - \langle \boldsymbol{\phi}, \boldsymbol{\phi}'\rangle) = 0$ with the variance bounded by $\frac{1}{K'}(\langle \boldsymbol{\phi}, \boldsymbol{\phi}'\rangle^2 + \|\boldsymbol{\phi}\|_2^2\|\boldsymbol{\phi}'\|_2^2)$.

4. Proposed Approach

Overview. Kernelized Few-shot Object Detector (KFSOD) is trained with a set of L-way Z-shot episodes. Each episode contains a query image with objects, and Z support regions (object crops) for each of L sampled classes. The training protocol ensures that query objects match some support objects by label. During testing, KFSOD localizes and classifies objects in the query image given annotated support crops of novel classes. KFSOD in Fig. 1 contains[3]:

1. Fig. 2a: Encoding Network (EN) yields conv. feature maps (stride along the channel mode is a feature vector).
2. Fig 3a &4: Kernelized Autocorrelation (KA) unit forms two types of kernelized representations: (i) RKHS kernels and (ii) so-called kernelized autocorrelation matrices called k-autocorrelations. In practice, KA computes both types of kernelization for the linear, polynomial and RBF kernelized non-linearities from support crops and query RoIs. As there are only a few support regions per episode, we crop support regions and directly kernelize them. We use a different approach (point 4) for query images which have many more RoIs.

[3]Detailed KFSOD pipeline (all modules) is in §B of **Suppl. Material**.

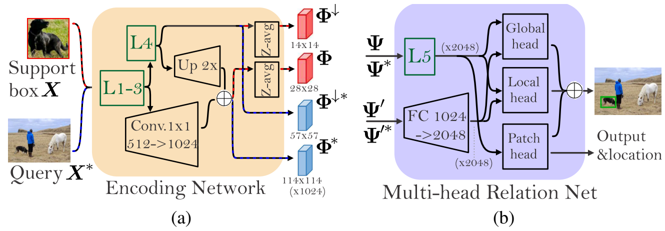


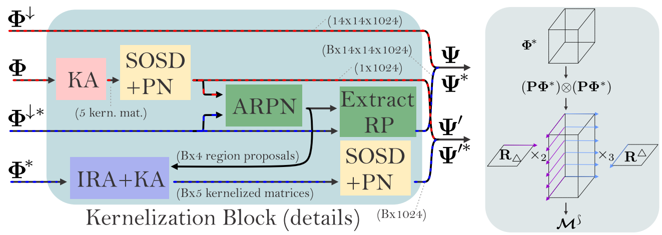
Figure 2. Our Encoding Network (EN) in Fig. 2a uses layers L1-4 of ResNet-50. Block (Up 2x) performs upsampling, $\oplus$ is addition. For simplicity, Z-shot samples are averaged per class by block (Z-avg). $\boldsymbol{\Phi}^{\downarrow}$ and $\boldsymbol{\Phi}$ are low and high-resolution feature maps. Multi-head Relation Net in Fig. 2b receives first-order representations ($\boldsymbol{\Psi}$ for support, $\boldsymbol{\Psi}^*$ for the query) and kernelized representations ($\boldsymbol{\Psi}'$ for support, $\boldsymbol{\Psi}'^*$ for the query) from the Kernelization Block. From first-order maps, layer L5 of ResNet-50 generates feature maps with 2048 channels. An FC layer maps representations of 1024 to 2048 dimensional space. Such feature maps are fed into the global, local and patch heads. (See §D in **Suppl. Material**.)

3. Fig. 3a: Attention Region Proposal Network (ARPN) takes kernelized feature vectors per support region to cross-correlate them against the image-wise query convolutional feature map to produce a query attention map. Region Proposal Network outputs query RoIs.
4. Fig. 3a, 3b &4: Integral RoI Aggregation (IRA) rapidly forms inner-product matrices required to obtain RKHS kernels and k-autocorrelations for each query RoI.
5. Fig 2b: Multi-head Relation Network (MRN) combines first-order representations (spatial-wise cues) with spatially-invariant kernelized representations to learn relations between support-query region pairs, and predict classes and locations of objects in query images.

We now describe these components in detail.

Encoding Network (Fig. 2a). The support crop and the query image are denoted as $\mathbf{X} \in \mathbb{R}^{W \times H}$ and $\mathbf{X}^* \in \mathbb{R}^{W^* \times H^*}$. Let $\boldsymbol{\Phi}^{\downarrow} \in \mathbb{R}^{K \times N}$ and $\boldsymbol{\Phi}^{\downarrow *} \in \mathbb{R}^{K \times N^*}$ be support and query maps from layer 4. Feature map $\boldsymbol{\Phi}^{\downarrow *} \in \mathbb{R}^{K \times N^*}$ is used by ARPN. Let $\boldsymbol{\Phi} \in \mathbb{R}^{K \times 4N}$ and $\boldsymbol{\Phi}^* \in \mathbb{R}^{K \times 4N^*}$ be feature maps with twice the resolution of $\boldsymbol{\Phi}^{\downarrow}$ and $\boldsymbol{\Phi}^{\downarrow *}$. They are used for forming RKHS kernels and k-autocorrelations. (See §A of **Suppl. Material** for details about EN.)

RKHS Kernels and K-autocorrelations. Before we generate our representations, we perform the ℓ_2-norm normalization on feature vectors $\boldsymbol{\Phi} \in \mathbb{R}^{K \times 4N}$. An autocorrelation matrix on $\boldsymbol{\Phi}$ could be then computed as $\mathbf{K}^{(\text{lin})} = \frac{1}{4N}\boldsymbol{\Phi}\boldsymbol{\Phi}^T$. Let $\bar{\boldsymbol{\Phi}} = \boldsymbol{\Phi}^T \equiv [\bar{\boldsymbol{\phi}}_1, \cdots, \bar{\boldsymbol{\phi}}_K]$, then one could also write $\mathbf{K}^{(\text{lin})}$ as an inner-product kernel $\mathbf{k}^{(\text{lin})}_{ij} = \langle \bar{\boldsymbol{\phi}}_i, \bar{\boldsymbol{\phi}}_j \rangle$. In practice, we firstly obtain **RKHS kernels** listed in Table 1, *e.g*., $\mathbf{k}^{(\text{poly})}_{ij}$ and $\mathbf{k}^{(\text{rbf})}_{ij}$, by substituting the inner product $\langle \bar{\boldsymbol{\phi}}_i, \bar{\boldsymbol{\phi}}_j \rangle$ into a non-linearity $\rho : \mathbb{R} \to \mathbb{R}$ in Table 2. In addition, for the RBF kernel, one should decompose the Euclidean distance $\|\bar{\boldsymbol{\phi}}_i - \bar{\boldsymbol{\phi}}_j\|_2^2 = \|\bar{\boldsymbol{\phi}}_i\|_2^2 + \|\bar{\boldsymbol{\phi}}_j\|_2^2 - 2\langle \bar{\boldsymbol{\phi}}_i, \bar{\boldsymbol{\phi}}_j \rangle$, which becomes handy during IRA-based computations.



(a) (b)

Figure 3. Kernelized Block (Fig. 3a). Feature Maps ($\mathbf{\Phi}^{\downarrow}$ and $\mathbf{\Phi}$ are of low and high resolution) enter the block. As support bounding boxes are few, RKHS kernels and k-autocorrelations are computed by the Kernelized Autocorrelation (KA) unit, and passed to the Second-order Spectral Diagonal Correlation (SOSD) unit with Power Normalization (PN). Attention Region Proposal Network (ARPN) matches support representations against the query feature map to obtain the query attention map and generate B query region proposals. The Integral RoI Aggregation (IRA) unit produces the so-called integral tensors for RKHS kernels and k-autocorrelations. ARPN passes region candidates to IRA, where RKHS kernels and k-autocorrelations are extracted for each candidate region at a low cost, and passed via the SOSD with PN. First-order and kernelized representations $\mathbf{\Psi}$ and $\mathbf{\Psi}'$ are passed to the Multi-head Relation Network (MRN). Fig. 3b: computation of the integral tensor in Eq. (5). Fibers (purple arrows) are multiplied with each other by the inner product. By analogy, blue fibers are multiplied with each other. Ignore count sketching matrix $\mathbf{P}$ for generic IRA. In fact, tensor $\mathcal{M}^{\int}$ can be obtained efficiently with two CUDA BLAS broadcast-style matrix-matrix multiplications.

Secondly, we define and compute a family of **k-autocorrelation** matrices. Let $\mathbf{\Phi}\equiv[\boldsymbol{\phi}_1,\cdots,\boldsymbol{\phi}_{4N}]$, then we can define a family of kernelized autocorrelations $k'_{ij}=\frac{1}{4N}\sum_{n\in\mathcal{I}_{4N}}\rho\big(\phi_{in}\phi_{jn}^T\big)$ with the equivalent matrix form $\mathbf{K}'=\frac{1}{4N}\sum_{n\in\mathcal{I}_{4N}}\rho\big(\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T\big)$. By substituting ρ according to Table 2, we obtain $k'^{(\text{poly})}_{ij}$ and $k'^{(\text{rbf})}_{ij}$. Note that $k'^{(\text{lin})}_{ij}=k^{(\text{lin})}_{ij}$.

	linear	polynomial	RBF
$\mathbf{k}_{ij}$	$\langle\bar{\boldsymbol{\phi}}_i,\bar{\boldsymbol{\phi}}_j\rangle$	$\big(\langle\bar{\boldsymbol{\phi}}_i,\bar{\boldsymbol{\phi}}_j\rangle+\lambda\big)^r$	$\exp\big(-\lVert\bar{\boldsymbol{\phi}}_i-\bar{\boldsymbol{\phi}}_j\rVert_2^2/2\sigma^2\big)$

Table 1. Kernels on vectors $\bar{\boldsymbol{\phi}}$. We learn hyper-parameters $\lambda\geq 0$ and $\sigma\geq 0$, whereas $r\in\mathcal{I}_3$ is the order of polynomial kernel. Note $\exp\big(-\lVert\bar{\boldsymbol{\phi}}_i-\bar{\boldsymbol{\phi}}_j\rVert_2^2/2\sigma^2\big)\propto\exp\big(\langle\bar{\boldsymbol{\phi}}_i,\bar{\boldsymbol{\phi}}_j\rangle/\sigma^2\big)$ if $\forall i,\lVert\bar{\boldsymbol{\phi}}_i\rVert_2=1$.

	linear	polynomial	RBF
$\rho(z)$	z	$(z+\lambda)^r$	$\exp\big(z/\sigma^2\big)$
$\boldsymbol{\omega}(x)$	x	$\big[\binom{r}{\iota}^{\frac{1}{2}}\lambda^{\frac{\iota}{2}}x^{\frac{r-\iota}{2}}\big]_{\iota=0,\cdots,r}$	$\big[\sqrt{c}\exp\big(\frac{2(x-q_\iota)}{\sigma^2}\big)\big]_{\iota\in\mathcal{I}_{r'}}$

Table 2. Non-linearity $\rho(z)$ can form RKHS kernels (substitute the inner product for z) and/or kernelized autocorrelation matrices. For approximations, we use feature maps *e.g.*, if $z=xy$ then $\rho(z)\simeq\langle\boldsymbol{\omega}(x),\boldsymbol{\omega}(y)\rangle$, where r is the polynomial order, $r'\approx 3$ determines the quality of RBF approximation, $c>0$ is a scaling constant and $q_1,\cdots,q_{r'}$ are equally spaced to cover interval $\langle -1,1\rangle$.

RKHS kernels *vs.* **k-autocorrelations**. For kernels, the inner product precedes the non-linearity ρ: $k_{ij}=\rho(\sum_l\bar{\phi}_{li}\bar{\phi}_{lj})$. The non-linearity acts on a sum-trend of element-wise correlations. In contrast, for k-autocorrelations, ρ, applied to individual coefficients, precedes the inner product: $k'_{ij}=\sum_l\rho(\bar{\phi}_{li}\bar{\phi}_{lj})$. It acts as a soft-maximum selector over individual element-wise correlation pairs.

Integral RoI Aggregation[4] **(IRA).** For query proposal regions, we have access to a feature map $\mathbf{\Phi}^*$ representing an entire query image. Instead of extracting RoIs (proposed by RPN) from $\mathbf{\Phi}^*$ and then forming hundreds of kernel matrices individually (*e.g.* 256 RoIs $\times$ 5 kernelized representations), we propose an efficient IRA (Fig. 4: blue block). We form a correlation feature map by the Kronecker product along the first mode of $\mathbf{\Phi}^*$, extracting the upper triangular[5], and reshaping the matrix into a three-mode feature map:

$$\boldsymbol{\mathcal{M}}=\text{Reshape}\Big(\text{Upper}(\mathbf{\Phi}^*\otimes\mathbf{\Phi}^*)\in\mathbb{R}^{\frac{1}{2}K(K+1)\times 4N^*}\Big),\quad(4)$$

where $\boldsymbol{\mathcal{M}}\in\mathbb{R}^{\frac{1}{2}K(K+1)\times 2N_W^*\times 2N_H^*}$ due to Reshape$(\cdot)$ reshaping mode 2 (size $4N^*$) into modes 2 and 3 (sizes $2N_W^*$ and $2N_H^*$). Then the so-called integral tensor is formed.

Integral tensor is obtained by multiplying $\boldsymbol{\mathcal{M}}$ along modes 2 and 3 by the lower and upper triangular matrices, $R_{\triangle,ij}=1$ if $i\geq j$ (0 otherwise) and $R^{\triangle}_{ij}=1$ if $i\leq j$ (0 otherwise), in order to obtain the integral tensor

$$\boldsymbol{\mathcal{M}}^{\int}=\boldsymbol{\mathcal{M}}\times_2\mathbf{R}_{\triangle}\times_3\mathbf{R}^{\triangle},\qquad(5)$$

where symbols $\times_2$ and $\times_3$ are tensor-tensor multiplications along modes 2 and 3 [14].

Figure 3b illustrates the formation of integral tensor by considering each channel of $\boldsymbol{\mathcal{M}}$ separately. Let $\boldsymbol{\mathcal{M}}\equiv[\boldsymbol{M}_l]_{l\in\mathcal{I}_{K(K+1)/2}}$ and $\boldsymbol{\mathcal{M}}^{\int}\equiv[\boldsymbol{M}^{\int}_l]_{l\in\mathcal{I}_{K(K+1)/2}}$. Then $\forall l,\boldsymbol{M}^{\int}_l=\mathbf{R}_{\triangle}\boldsymbol{M}_l\mathbf{R}^{\triangle}$, or simply, $M_{lij}=\sum_{i'\leq i,j'\leq j}M_{li'j'}$.

Kernelized region representation with top-left (x,y) and bottom-right (x',y') locations, $x\leq x',y\leq y'$, is extracted with a negligible cost (one addition/two subtractions) by:

$$\mathbf{K}^{(\rho)}((x,y),(x',y'))=\zeta\rho(\hat{\boldsymbol{K}})\qquad\text{where}\qquad(6)$$
$$\hat{\boldsymbol{K}}=\text{Deploy}(\boldsymbol{\mathcal{M}}^{\int}_{:,x',y'}-\boldsymbol{\mathcal{M}}^{\int}_{:,x',y-1}-\boldsymbol{\mathcal{M}}^{\int}_{:,x-1,y'}+\boldsymbol{\mathcal{M}}^{\int}_{:,x-1,y-1}).$$

Deploy$(\cdot)$ in Eq. (6) deploys the vector containing the upper triangular (plus diagonal) back into the corresponding

[4]Viola and Jones [44] explain basics of integral images.

[5]Note that $\boldsymbol{\phi}\otimes\boldsymbol{\phi}=\text{Vec}(\boldsymbol{\phi}\boldsymbol{\phi}^T)\in\mathbb{R}^{K^2}$ where Vec$(\cdot)$ vectorizes the matrix. We apply $\otimes$ to feature vectors $\boldsymbol{\phi}$ of $\mathbf{\Phi}^*$ in Eq. (4). We discard redundant lower triangular by Upper$(\boldsymbol{\phi}\otimes\boldsymbol{\phi})\equiv$ Upper$(\boldsymbol{\phi}\boldsymbol{\phi}^T)\in\mathbb{R}^{K(K+1)/2}$ that extracts the upper triangular+diagonal from the matrix & vectorizes it.

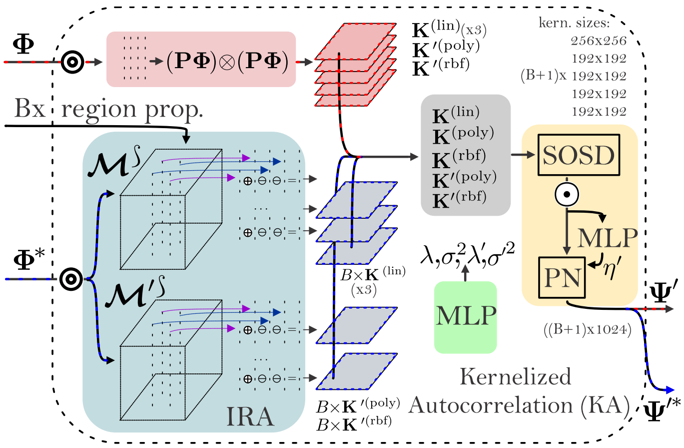


Figure 4. The KA unit takes the support and query feature maps $\mathbf{\Phi}$ and $\mathbf{\Phi}^*$ as input. Operator $\circledcirc$ splits $\mathbf{\Phi}$ and $\mathbf{\Phi}^*$ into 5 groups along the channel mode (1×256 and 4×192 groups on which 5 kernelizations are computed). For support, KA computes 3 RKHS linear kernels $\mathbf{K}^{(\text{lin})}$, two of which are used to form $\mathbf{K}^{(\text{poly})}$ and $\mathbf{K}^{(\text{rbf})}$ by applying relevant non-linearities. KA also computes k-autocorrelations $\mathbf{K}'^{(\text{poly})}$ and $\mathbf{K}'^{(\text{rbf})}$. For the query feature map $\mathbf{\Phi}^*$, IRA computes integral tensors $\boldsymbol{\mathcal{M}}^{\int}$ and $\boldsymbol{\mathcal{M}}'^{\int}$ from which $B\times 3$ RKHS linear kernels, $B\times$ polynomial and $B\times$ RBF k-autocorrelation matrices are cheaply extracted for B query proposal regions. $B\times$ RKHS polynomial and $B\times$ RBF kernels are computed from $B\times$ RKHS linear kernels. We obtain $(B+1)\times 5$ matrices that are kernel-pooled by SOSD (Table 3) to form signatures (1×256 and 4×192) that are concatenated by $\odot$ and passed via PN. The output of KA is $\mathbf{\Psi}'$ (one vector) and $\mathbf{\Psi}'^*$ (B vectors).

square matrix, $\zeta=\frac{1}{(x'-x+1)(y'-y+1)}$ is a normalization, whereas ':' simply extracts the entire fiber (vector) along the first mode of tensor. For the RBF kernels, we normalize $\hat{\mathbf{K}}$ by ℓ_2 norms of $\bar{\phi}_i$:

$$\hat{\mathbf{K}} := \text{Diag}^{\dagger}(\text{Diag}(\hat{\mathbf{K}}^{0.5}))\hat{\mathbf{K}}\text{Diag}^{\dagger}(\text{Diag}(\hat{\mathbf{K}}^{0.5})). \qquad (7)$$

For **k-autocorrelation matrices**, we have:

$$\boldsymbol{\mathcal{M}}'=\text{Reshape}(\text{Upper}(\rho(\mathbf{\Phi}^*)\otimes\rho(\mathbf{\Phi}^*))), \qquad (8)$$

with the integral tensor $\boldsymbol{\mathcal{M}}'^{\int}=\boldsymbol{\mathcal{M}}'\times_2\mathbf{R}_{\triangle}\times_3\mathbf{R}^{\triangle}$, so

$$\mathbf{K}'^{(\rho)}((x,y),(x',y'))=\zeta\hat{\boldsymbol{K}}' \qquad \text{where} \qquad (9)$$
$$\hat{\boldsymbol{K}}'=\text{Deploy}\big(\boldsymbol{\mathcal{M}}'^{\int}_{:,x',y'}-\boldsymbol{\mathcal{M}}'^{\int}_{:,x',y-1}-\boldsymbol{\mathcal{M}}'^{\int}_{:,x-1,y'}+\boldsymbol{\mathcal{M}}'^{\int}_{:,x-1,y-1}\big).$$

In experiments, we use (i) RKHS-based linear, polynomial and RBF kernels, and (ii) k-autocorrelations with polynomial and RBF non-linearities. We split K support feature maps $\mathbf{\Phi}$ and query[6] feature maps $\mathbf{\Phi}^*$ along the channel mode into 5 equally sized groups (details provided later). Such a setting limits computational cost and lets each kernel specialize.

[6]Due to the low number of support regions, their kernelized representations are computed by the basic formula. For query proposal regions, we use IRA-based computations due to the large number of proposals $\geq$256.

Integral RoI Aggregation with Count Sketching. Eq. (4) and (8) apply the costly Kronecker product on K dimensional feature vectors. To reduce its $\mathcal{O}(K^2)$ complexity to $\mathcal{O}(K'^2)$, $K'\ll K$, we apply count sketching [47] via the unitary projection matrix $\mathbf{P}\in\{-1,0,1\}^{K'\times K}$ which enjoys a simple pseudoinverse $\mathbf{P}^{\dagger}=\frac{K'}{K}\mathbf{P}^T$. Eq. (4) and (8) become

$$\boldsymbol{\mathcal{M}}=\text{Reshape}\Big(\text{Upper}\Big((\mathbf{P\Phi}^*)\otimes(\mathbf{P\Phi}^*)\Big)\Big) \quad \text{and} \qquad (10)$$
$$\boldsymbol{\mathcal{M}}'=\text{Reshape}\Big(\text{Upper}\Big(\sum_{\iota\in\mathcal{I}_{r'}}(\mathbf{P}\omega_{\iota}(\mathbf{\Phi}^*))\otimes(\mathbf{P}\omega_{\iota}(\mathbf{\Phi}^*))\Big)\Big),$$

with the kernelized region representations recovered by

$$\mathbf{K}^{(\rho)}((x,y),(x',y'))\approx\zeta\rho(\mathbf{P}^{\dagger}\hat{\mathbf{K}}\mathbf{P}^{\dagger T}), \qquad (11)$$
$$\mathbf{K}'^{(\rho)}((x,y),(x',y'))\approx\zeta\mathbf{P}^{\dagger}\hat{\mathbf{K}}'\mathbf{P}^{\dagger T}. \qquad (12)$$

The above is true as $\mathbf{\Phi\Phi}^T\approx\mathbf{P\Phi\Phi}^T\mathbf{P}^T$ so inverting sketching yields $\mathbf{P}^{\dagger}\mathbf{P\Phi\Phi}^T(\mathbf{P}^{\dagger}\mathbf{P})^T\approx\mathbf{\Phi\Phi}^T$.

Count sketching implicitly performs a feature-level augmentation akin to injecting the Gaussian noise into features [46]. (Proof is in §C of **Suppl. Material**.)

•Notice $\langle\mathbf{P}\phi,\mathbf{P}\phi'\rangle=\langle\mathbf{P}^T\mathbf{P}\phi,\phi'\rangle=\langle\frac{K}{K'}\mathbf{P}^{\dagger}\mathbf{P}\phi,\phi'\rangle$. If $\|\phi\|_2=\|\phi'\|_2=1$ then based on properties of count sketches in §3, we have

$$\big\langle\tfrac{K}{K'}\mathbf{P}^{\dagger}\mathbf{P}\phi,\phi'\big\rangle=\langle\mathbf{P}\phi,\mathbf{P}\phi'\rangle\sim\mathcal{N}\big(\langle\phi,\phi'\rangle,\sigma^{\dagger 2}\big), \qquad (13)$$

where $\langle\phi,\phi'\rangle$ is a point-wise convolution of feature vector ϕ with convolutional filter ϕ'. It follows that $\langle\frac{K}{K'}\mathbf{P}^{\dagger}\mathbf{P}\phi,\phi'\rangle$ realizes a point-wise noisy convolution whose variance $\sigma^{\dagger 2}=\frac{1}{K'}(\langle\phi,\phi'\rangle+1)\leq\frac{2}{K'}$.

•Injecting the Gaussian noise [46] can be characterized as $\langle\phi+\Delta\phi,\phi'\rangle$ where $\Delta\phi\sim\mathcal{N}(\mathbf{0},\sigma^{\ddagger 2})$ which leads to $\langle\phi+\Delta\phi,\phi'\rangle\sim\mathcal{N}\big(\langle\phi,\phi'\rangle,\sigma^{\ddagger 2}\big)$.

Computational Complexity of IRA. Computing dot-product based kernels naively (from $\mathbf{\Phi}^*$) has the complexity $\mathcal{O}(K^2\tilde{N}^*B)$, where K is the number of features (channels) and $\tilde{N}^*$ is the average area of $B=256$ query proposals. Computing these kernels via IRA has the complexity $\mathcal{O}(K^2N^{*\frac{3}{2}}+K^2B)$ (the first and second terms concern forming the integral tensor and extracting B kernels from it). If $N^*\ll(\tilde{N}^*B)^{\frac{2}{3}}$, computing kernels via IRA is faster.

The cost of computing the Kronecker product in Eq. (4) is $\mathcal{O}(K^2N^*)$ which reduces to $\mathcal{O}(KK'N^*+K'^2N^*)$ for the sketching variant in Eq. (10) (top) and $\mathcal{O}(KK'N^*+r'K'^2N^*)$ (bottom). If $K'=0.5K$, the cost is reduced $4\times$.

Pooling Kernelized Representations. Using matrices $\mathbf{K}$ of dimensions $K\times K$ as feature representations is prohibitively expensive given hundreds of RoIs. Thus, we pool $\mathbf{K}$ into a K-dimensional representation. Table 3 lists three pooling operators we consider: first-order (mean) pooling with PN, denoted as FO+PN, Second-order Spectral Diagonal Correlation with PN (SOSD+PN), and Second-order Self Correlation with PN (SOSC+PN). We use SOSD+PN (standard second-order pooling) and ablate other operators in Sec. 5.

Extraction of RKHS Kernels and K-autocorrelations. As shown in Figure 4, given B query RoIs and the feature maps for a support crop ($\mathbf{\Phi}$) and query image ($\mathbf{\Phi}^*$), we generate 5 kernelized matrices from $\mathbf{\Phi}$, and $B\times 5$ kernelized matrices from $\mathbf{\Phi}^*$. Each set of 5 kernelized matrices correspond to 3 (linear, polynomial and RBF) RKHS kernels, and 2 (polynomial and RBF) k-autocorrelation matrices. Each kernelized matrix is computed on 1 of 5 groups of channel features, where the groups were created by splitting (denoted by $\circledcirc$ in Figure 4) the channel with size $K = 1024$ into 1 group with size 256 and 4 groups each with size 192.

To set λ and σ^2, the hyper-parameters of polynomial and RBF RKHS kernels, we predict them using a trained layer $\text{MLP}(\mathbf{K}^{(\text{lin})}\cdot\mathbf{1})$, where MLP contains an FC layer followed by the sigmoid function. For k-autocorrelations, we first take the mean over the spatial modes of $\mathbf{\Phi}$ and $\mathbf{\Phi}^*$. We split the resulting vectors $\boldsymbol{\mu}$ and $\boldsymbol{\mu}^*$ into groups, as explained above, and feed them into an MLP to generate σ'^2 and λ'. Parameter η' of PN in Table 3 is predicted by a different MLP that uses the output of SOSD as its input. As sigmoid outputs are in range $\langle 0,1\rangle$, for RBF kernels, polynomial kernels and PN, we scale them into $\langle 0.1,2\rangle$, $\langle 1,10\rangle$ and $\langle 1,10^3\rangle$ ranges. Figure 4 shows the extraction procedure.

Multi-head Relation Network. MRN learns the similarity score, top-left and bottom-right bounding box coordinates between support-query pairs represented by (i) first-order spatially-aware $(\mathbf{\Psi},\{\mathbf{\Psi}^*_b\}_{b\in\mathcal{I}_B})$ which are feature maps in $\mathbb{R}^{1024\times14\times14}$, and (ii) second-order spatially-invariant pooled kernelized representations $(\mathbf{\Psi}',\{\mathbf{\Psi}'^*_b\}_{b\in\mathcal{I}_B})$ which are vectors in $\mathbb{R}^{1024}$. Notice we have B query candidate regions.

The Multi-head Relation Net in Figure 2b contains 3 subheads: (i) global head, (ii) local head and (iii) patch head. Global and local heads combine first-order spatially-aware and second-order spatially-invariant representations to learn the similarity. The patch head takes first-order spatially-aware representations to perform the bounding box regression. See §D of **Suppl. Material** for more details.

Let $s((\mathbf{\Psi},\{\mathbf{\Psi}^*_b\}_{b\in\mathcal{I}_B}),(\mathbf{\Psi}',\{\mathbf{\Psi}'^*_b\}_{b\in\mathcal{I}_B});\mathcal{S})\!\rightarrow\!\{(\bar{\mathbf{y}},\bar{\mathbf{x}})_b\}_{b\in\mathcal{I}_B}$. $\mathcal{S}$ are network parameters, $(\bar{\mathbf{y}},\bar{\mathbf{x}})\in\mathcal{Y}$ contains similarity prediction, and top-left/bottom-right coordinates of candidate regions $b\in\mathcal{I}_B$. For the L-way Z-shot problem, we have $L\times Z$ support image regions $\{\mathbf{X}_n\}_{n\in\mathcal{U}}$ from set $\mathcal{U}$ and their corresponding descriptors $\{(\mathbf{\Phi}^{\downarrow},\mathbf{\Phi})_n\}_{n\in\mathcal{U}}$ from the Encoding Network (Fig. 2a). We compute L representations $\mathbf{\Psi}$ and $\mathbf{\Psi}'$. Let $\mathbf{X}^*$ be a query image with its query feature maps $\{(\mathbf{\Phi}^{\downarrow*},\mathbf{\Phi}^*)_n\}_{n\in\mathcal{I}_B}$ obtained from B proposal regions from ARPN (Fig. 3a). Representations $(\mathbf{\Phi}^{\downarrow},\mathbf{\Phi})$ and $(\mathbf{\Phi}^{\downarrow*},\mathbf{\Phi}^*)$ are passed to the Kernelized Block (Fig. 3) whose output representations $(\mathbf{\Psi},\mathbf{\Psi}')$ and $(\mathbf{\Psi}^*,\mathbf{\Psi}'^*)$ are passed to Multi-head Relation Net to minimize the loss:

$$\sum_{(l,b)\in\mathcal{I}_L\times\mathcal{I}_B} l_{sim}(\bar{\mathbf{y}}^l_b,\mathbf{y}^l_b)+l_{box}(\bar{\mathbf{x}}^l_b,\mathbf{x}^l_b)+l^{\mathcal{H}}_{rpn}(\mathbf{\Psi}'_l\times\mathbf{\Phi}^{\downarrow*},\mathbf{\Phi}^{\downarrow*}),\quad (14)$$

where query-support pairs belong to L classes in the subset $C^{\ddagger}\equiv\{c_1,\cdots,c_L\}\subset\mathcal{I}_C\equiv\mathcal{C}$. Loss functions l_{box} and $l^{\mathcal{H}}_{rpn}$ follow [31], and l_{sim} is the binary cross-entropy, $\mathcal{H}$ are parameters of ARPN (same with [31]) and '$\times$' performs channel-wise cross-correlation in $\mathbf{\Psi}'_l\times\mathbf{\Phi}^{\downarrow*}\in\mathbb{R}^{1024\times14\times14}$.

FO+PN	SOSD+PN	SOSC+PN
$\mathcal{G}_{\text{SigmE}}\left(\frac{\mathbf{\Phi}^{\downarrow}\mathbf{1}}{N};\eta'\right)$	$\mathcal{G}_{\text{SigmE}}\left(\text{Diag}\left(\widehat{\mathcal{G}}_{\text{MaxExp}}(\mathbf{K};\eta)\right);\eta'\right)$	$\mathcal{G}_{\text{SigmE}}\left(\frac{\mathbf{K}\mathbf{1}}{K};\eta\right)$

Table 3. Pooling operators are applied (i) prior to cross-correlation in ARPN and (ii) to represent support and query RoIs.

5. Experiments

Datasets and settings. For PASCAL VOC 2007/12 [5], we adopt the 15/5 base/novel category split setting and use training/validation sets from PASCAL VOC 2007 and 2012 for training, and the testing set from PASCAL VOC 2007 for testing [11]. For MS COCO [24], we follow [52], and adopt the 20 categories that overlap with PASCAL VOC as the novel categories (testing). The remaining 60 categories are used for training. For the FSOD dataset [6], we split 1000 categories into 800/200 for training/testing. We report standard FSOD metrics: mAP, AP, AP_{50} and AP_{75}.

Implementation details are in §H and hyper-parameters for each dataset are in §I of **Suppl. Material**.

5.1. Comparisons with the State of the Art

PASCAL VOC 2007/12. We compare KFSOD to FSOD$^{\text{up}}$ [48], CGDP+FRCN [23], TIP [21], FSCE [40], TFA [45], Feature Reweighting (FR) [11], LSTD [2], FRCN [31], NP-RepMet [53], MPSR [49], PSND [58] and FSOD [6]. Table 4 shows that KFSOD outperforms FSOD by a 6.3–10% margin. For the 1- and 10-shot regime, we outperform FSOD$^{\text{up}}$ by $\sim$2.2%. Table 9 (§E of **Suppl. Material**) shows class-wise results (5-shot protocol): KFSOD gains 11.2% and 6.5% mAP (novel and base classes) over FSOD.

MS COCO. Table 5a compares KFSOD *vs.* FSOD$^{\text{up}}$ [48], CGDP+FRCN [23], TIP [21], FSCE [40], TFA [45], FR [11], Meta R-CNN [52], FSOD [6] and PNSD [6] on MS COCO minival set (20 novel cat., 10-shot). KFSOD outperforms FSOD$^{\text{up}}$ by 6.9%, 2.4%, 8.9% (AP, AP_{50}, AP_{75}).

FSOD. In Table 5b we compare KFSOD (5-shot protocol) with PNSD [58], FSOD [6], LSTD [2] and LSTD (FRN [31]). We re-implement BD&TK, modules of LSTD, based on Faster-RCNN for a fair comparison. KFSOD gives SOTA results of 33.4% AP_{50} and 29.6% AP_{75}.

5.2. Ablation studies

Below we use PASCAL VOC (novel classes, split 1, 5-shot setting, hyper-parameters selected on the val. split).

Table 4. Comparison of different methods in terms of mAP (%) on three splits on the VOC 2007 testing set.

Method/Shot		Split 1				Split 2				Split 3				Mean±std			
		1	3	5	10	1	3	5	10	1	3	5	10	1	3	5	10
FRCN	ICCV12	11.9	29.0	36.9	36.9	5.9	23.4	29.1	28.8	5.0	18.1	30.8	43.4	7.6±3.1	23.5±4.5	32.3±3.3	36.4±6.0
FR	ICCV19	14.8	26.7	33.9	47.2	15.7	22.7	30.1	39.2	19.2	25.7	40.6	41.3	16.6±1.9	25.0±1.7	34.9±4.3	42.6±3.4
Meta	ICCV19	19.9	35.0	45.7	51.5	10.4	29.6	34.8	45.4	14.3	27.5	41.2	48.1	14.9±3.9	30.7±3.2	40.6±4.5	48.3±2.5
FSOD	CVPR20	37.8	48.7	55.5	58.2	28.9	40.7	42.1	47.6	28.6	38.1	44.7	47.5	29.5±1.0	40.9±5.6	45.5±3.2	47.8±5.0
NP-RepMet	NeurIPS20	37.8	41.7	47.3	49.4	**41.6**	43.4	47.4	49.1	33.3	39.8	41.5	44.8	37.6±3.4	41.6±1.5	45.4±2.8	47.8±2.1
PNSD	ACCV20	40.9	50.4	56.5	59.8	30.2	41.8	46.4	48.3	34.8	40.6	46.9	48.6	35.3±4.4	44.3±4.4	48.6±2.8	52.2±5.4
MPSR	ECCV20	41.7	51.4	55.2	61.8	24.4	39.2	39.9	47.8	**35.6**	42.3	48.0	49.7	33.9±7.2	44.3±5.2	47.7±6.2	53.1±6.2
TFA	ICML20	39.8	44.7	55.7	56.0	23.5	34.1	35.1	39.1	30.8	42.8	49.5	49.8	31.4±6.7	40.5±4.6	46.8±8.6	48.3±7.0
FSCE	CVPR21	44.2	51.4	**61.9**	63.4	27.3	**43.5**	44.2	50.2	22.6	39.5	47.3	**54.0**	31.4±9.3	44.8±4.9	51.1±7.7	55.9±5.6
CGDP+FRCN	CVPR21	40.7	46.5	57.4	62.4	27.3	40.8	42.7	46.3	31.2	43.7	50.1	55.6	33.1±5.6	43.67±2.3	50.0±6.0	54.8±6.6
TIP	CVPR21	27.7	43.3	50.2	56.6	22.7	33.8	40.9	46.9	21.7	38.1	44.5	50.9	24.0±2.6	38.4±4.0	45.2±4.3	52.47±5.3
FSOD^{up}	ICCV21	43.8	50.3	55.4	61.7	31.2	41.2	44.2	48.3	35.5	43.9	50.6	53.5	36.8±5.2	45.1±3.8	50.1±4.6	54.5±5.5
KFSOD	(Ours)	**44.6**	**54.4**	60.9	**65.8**	37.8	43.1	**48.1**	**50.4**	34.8	**44.1**	**52.7**	53.9	**39.1±3.8**	**47.2±5.1**	**53.9±3.6**	**56.7±6.0**

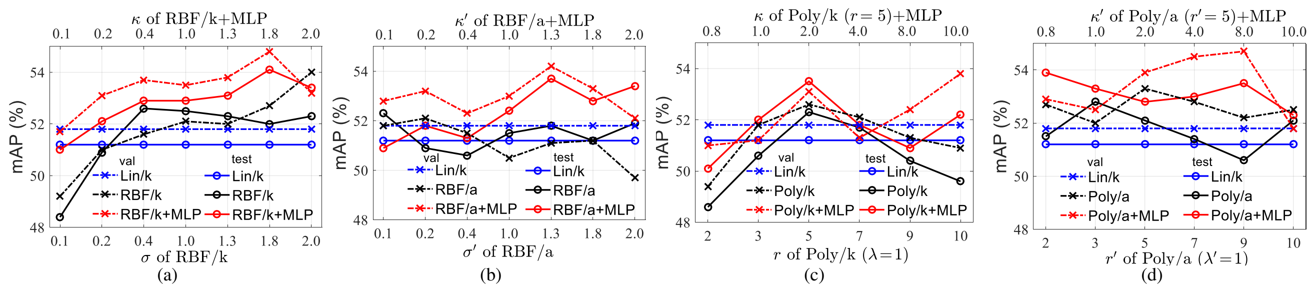


Figure 5. mAP% of individual kernelized representations (PASCAL VOC 2007, novel classes, split 1, 5-shot setting) w.r.t. kernel hyper-parameters. Fig. 5a shows ablations for the RKHS-based RBF kernels (σ of RBF/k and κ for RBF/k+MLP where κ is the sigmoid scaling parameter in the learnable MLP module that adjusts σ on the fly). Lin/k is the linear kernel (no hyper-parameters). Fig. 5b shows ablations for the k-autocorrelations (σ' of RBF/a and the sigmoid scaling parameter κ' for RBF/a+MLP). Fig. 5c shows ablations for the RKHS polynomial kernels (order r of Poly/k with the offset $\lambda=1$, and κ for Poly/k+MLP ($r=5$) where κ is the sigmoid scaling of MLP that adjusts λ on the fly). Fig. 5d: k-autocorrelations (order r' of Poly/a ($\lambda'=1$) and sigmoid scaling κ' of Poly/a+MLP ($r'=5$)).

Table 5. Comparison with SOTA on the MS COCO minival set and FSOD testing set, given in Tables 5a and 5b.

Shot	Method		AP	AP_{50}	AP_{75}
10	LSTD	AAAI18	3.2	8.1	2.1
	FR	ICCV12	5.6	12.3	4.6
	Meta	ICCV19	8.7	19.18	6.6
	MPSR	ECCV20	9.8	17.9	9.7
	FSOD	CVPR20	11.1	20.4	10.6
	PNSD	ACCV20	15.3	21.7	12.5
	TFA	ICML20	9.6	10.0	9.3
	FSCE	CVPR21	10.7	11.9	10.5
	CGDP+FRCN	CVPR21	11.3	20.3	11.5
	FSOD^{up}	ICCV21	11.6	23.9	9.8
	KFSOD	(Ours)	**18.5**	**26.3**	**18.7**

(a)

Shot	Method		AP_{50}	AP_{75}
5	LSTD (FRN)	AAAI18	23.0	12.9
	LSTD	AAAI18	24.2	13.5
	FSOD	CVPR20	27.5	19.4
	PNSD	ACCV20	29.8	22.6
	KFSOD	(Ours)	**33.4**	**29.6**

(b)

Table 6. Results with combinations of kernelized representations on FSOD and COCO dataset (5/10-shot protocol) are in Table 6a. Table 6b shows mAP on PASCAL VOC 2007 (5-shot, novel classes) of the linear kernel formed from low resolution $\mathbf{\Phi}^{\downarrow}$ *vs.* high-resolution $\mathbf{\Phi}$ feature maps (Fig. 2a).

Kernel			5-shot (FSOD)		10-shot (COCO)	
Lin/k	RBF/k+a	Poly/k+a	AP_{50}	AP_{75}	AP_{50}	AP_{75}
✓			30.6	23.8	21.3	12.4
	✓	✓	31.9	25.7	22.1	13.6
✓	✓	✓	**32.2**	**26.8**	**25.7**	**14.6**

(a)

B	5-shot (Novel) $\mathbf{\Phi}^{\downarrow}$	$\mathbf{\Phi}$
64	54.7	55.6
128	55.9	57.4
256	58.5	**59.3**
512	56.6	57.3
756	54.1	56.4
1024	53.4	55.2

(b)

Different backbones. See §G of **Suppl. Material**.

Performance of individual kernels. Firstly, we compare RKHS kernels with k-autocorrelations in a manual hyper-parameter setting. We set $\eta=5$ and $\eta'=100$ of SOSD+PN kernel pooling. We denote linear, RBF and polynomial RKHS kernels as Lin/k, RBF/k and Poly/k. We refer to RBF and polynomial k-autocorrelations as RBF/a and Poly/a. Fig. 5a shows that RBF/k ($\sigma=2.0$) outperforms Lin/k by 2.1% on the validation split. However, RBF/k+MLP ($\kappa=1.8$) outperforms RBF/k by 2% in both validation and test splits. Fig. 5b shows that RBF/a ($\sigma'=0.2$) outperforms Lin/k by ~0.2%. However, RBF/a+MLP with the sigmoid scaling parameter $\kappa'=1.3$ outperforms Link/k by ~2% on both validation and test splits. Fig. 5c shows that Poly/k ($r=5$) outperforms Link/k on both validation and test splits. Finally, Fig. 5d shows that Poly/a+MLP ($\kappa'=8$) outperforms Poly/a ($r'=5$) and Lin/k. Fig. 5b & 5c show that MLP adjusts parameters of kernels on the fly in a very stable way (maxima match on val. and testing splits). On FSOD and COCO (Table 6a), combining kernels improves results over Lin/k by ~3%. See §F of **Suppl. Material** for ablations on combinations of kernels.

Kernel pooling. We evaluate pooling from Table 3 w.r.t. η of Spectral Power Normalization (SPN) and η' of element-wise PN, which we learn by MLP with sigmoid scaling κ''.

Figure 6 shows that setting $\eta=1$ (SPN is switched-off) results in a big performance drop on all kernels ($\eta=1$ equals merely applying kernel non-linearities along the diagonals

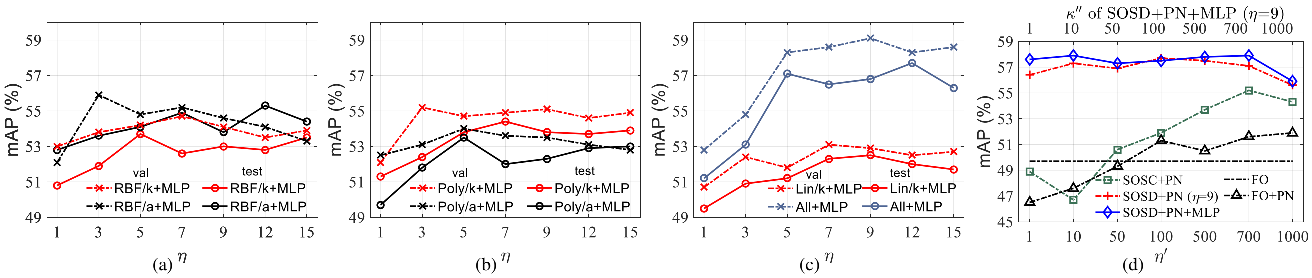


Figure 6. mAP % of individual kernelized representations (PASCAL VOC 2007, novel classes, split 1, 5-shot setting) w.r.t. η or η'. Fig. 6a: RBF/k+MLP and RBF/a+MLP, and Fig. 6b: Poly/k+MLP and Poly/a+MLP w.r.t. the SOSD+PN kernel pooling parameter η (η' is learnt on the fly). Fig. 6c shows the performance of all 5 kernelized representations w.r.t. η of SOSD+PN. Fig. 6d compares First Order (FO), First Order with PN (FO+PN), kernel pooling SOSC+PN and SOSD+PN, and SOSD+PN with η' adjusted by an MLP scaled by κ''.

Table 7. In Tab. 7a is the runtime (PASCAL VOC 2007) in seconds per 1000 images (training *vs*. inference time) w.r.t. B (the number of proposal regions). No count sketching was used. In Tab. 7b is the runtime in seconds per 1000 images. Count sketching was used by IRA+KA. Compression ratio $\frac{K}{K'}$ that gave best results was set to $2\times$, $4\times$ and $8\times$ on PASCAL VOC 2007, FSOD, and COCO.

B	Training IRA+KA	Training KA	Inference IRA+KA	Inference KA
64	95.7	98.6	56.2	57.6
128	104.4	140.0	61.5	68.3
256	127.6	164.7	70.1	87.2
512	153.6	208.4	81.8	117.8
756	229.2	325.6	87.8	152.9

(a)

Dataset (B=256)	Training IRA+KA	Training KA	Inference IRA+KA	Inference KA
PASCAL (VOC2007)	109.2	164.7	60.0	87.2
FSOD	101.1	171.0	55.8	88.3
COCO	69.7	172.1	39.0	89.6

(b)

Table 8. First-order *vs*. kernelized features in ARPN and/or MRN (mAP%, PASCAL VOC 2007, 5/10-shot, novel/base classes).

HARPN	MRN	Shot/Novel 5	Shot/Novel 10	Shot/Base 5	Shot/Base 10
FO	FO	49.6	57.2	65.7	68.8
FO	All+FO	52.3	60.4	67.9	71.3
All	All	59.8	64.2	72.1	75.4
All	All+FO	**60.9**	**65.8**	**72.6**	**76.6**

(a)

HARPN	MRN	Shot/Novel 5	Shot/Novel 10	Shot/Base 5	Shot/Base 10
All	FO	56.0	59.5	68.0	71.3
	PNSD	56.9	61.7	70.3	72.3
	Lin/k	57.8	62.9	71.2	73.0
	All	59.8	64.2	72.1	75.4

(b)

of matrices which discards off-diagonal correlations).

Fig.6a shows that the best combined validation performance is attained by both RKHS-based RBF/k+MLP and k-autocorrelation RBF/a+MLP for either $\eta = 5$ or $\eta = 9$. Fig. 6b also shows that if we combine the validation performance of Poly/k+MLP and Poly/a+MLP, either $\eta = 5$ or $\eta = 9$ are good choices. Thus we set $\eta = 9$ in all our experiments. Fig. 6c verifies the choice of $\eta = 9$ on Lin/k and the combination of all 5 kernelized representations.

Fig. 6d evaluates pooling operators from Table 3. Manually setting η' of FO+PN outperforms FO by $\sim$2%. Using all kernels with SOSC+PN pooling outperforms FO by $\sim$5%. SOSD+PN+MLP with η' adjusted by the MLP with κ'' outperforms FO by $\sim$8%. Table 8 ablates first-order *vs*. kernelized features. ARPN uses first-order+PN (FO) or all kernels (All). MRN head uses only first-order inputs $\mathbf{\Psi}, \mathbf{\Psi}^*$ (FO), kernelized inputs $\mathbf{\Psi}', \mathbf{\Psi}'^*$(All), linear kernel (Lin/k) or PSND [58] (second-order matrix).

Performance *vs*. speed (IRA+KA). In Fig. 7a, for KFSOD and the injection of Gaussian noise, we chose the best

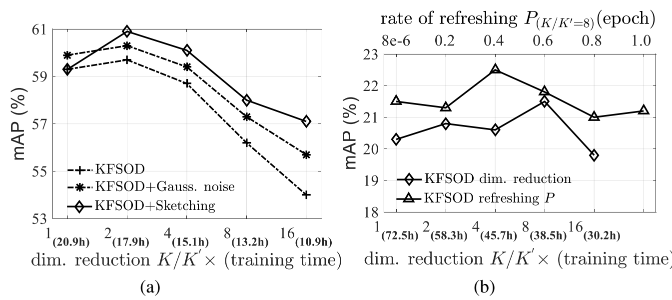


Figure 7. Performance (mAP %) of KFSOD, KFSOD+Gauss. noise and KFSOD+Sketching w.r.t. the compression ratio is in Fig. 7a (PASCAL VOC 2007, split 1, novel classes, 5-shot). mAP % of KFSOD with count sketching w.r.t. the frequency of drawing of sketch matrix $\mathbf{P}$ is in Fig. 7b (COCO, novel classes, 10-shot).

$\sigma^{\ddagger 2}$ (defined below Eq. (13)) in range $\langle 0.001, 0.1\rangle$, and we reduce the feature map size by the factor of K/K' indicated on x-axis (K' is not used). For KFSOD+Sketching, we set $\frac{K}{K'}$ (Section 3 defines K'). We indicate time in hours (all methods achieve similar speed-up) on x-axis. As is clear, KFSOD+Sketching achieves gain of $\sim$**1.8**% over KFSOD. Fig. 7b shows that the best performance of KFSOD+Sketching on COCO is achieved for drawing a new sketch matrix $\mathbf{P}$ every 0.4 epoch, and the best compression ratio is $\frac{K}{K'}=8$. Table 7a shows that IRA is very beneficial as B (the number of candidate regions) grows. Table 7b shows almost 3× speed-up compared to naive kernel computations on COCO.

6. Conclusions

We have proposed RKHS kernels and k-autocorrelations into FSOD. In order to accelerate the computation of region representations (the number of query candidate regions is large), we propose a novel Integral Region-of-Interest Aggregation (IRA) scheme combined with count sketching. On COCO, KFSOD with IRA and count sketching takes 38.5h instead of 85.2h. In addition to accelerating the computation of region representations, count sketching performs controlled feature augmentation due to its bounded noise, akin to feature augmentation by noise injection, leading to improved detection accuracy.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV 2020*, pages 213–229. Springer, 2020. 2

[2] Hao Chen, Yali Wang, Guoyou Wang, and Yu Qiao. LSTD: A low-shot transfer detector for object detection. In *AAAI 2018*, pages 2836–2843. AAAI Press, 2018. 1, 6

[3] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. UP-DETR: unsupervised pre-training for object detection with transformers. *CoRR*, abs/2011.09094, 2020. 2

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *CVPR 2009*, pages 248–255. IEEE Computer Society, 2009. 13

[5] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *Int. J. Comput. Vis.*, 88(2):303–338, 2010. 6

[6] Qi Fan, Wei Zhuo, and Yu-Wing Tai. Few-shot object detection with attention-rpn and multi-relation detector. *CoRR*, abs/1908.01998, 2019. 1, 2, 6

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML 2007*, pages 1126–1135. PMLR, 2017. 2

[8] Ross B. Girshick. Fast R-CNN. In *ICCV 2015*, pages 1440–1448. IEEE Computer Society, 2015. 1, 2

[9] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(8):2011–2023, 2020. 2

[10] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. On the burstiness of visual elements. In *CVPR*, 2009. 2

[11] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-shot object detection via feature reweighting. In *ICCV 2019*, pages 8419–8428. IEEE, 2019. 1, 2, 6

[12] Leonid Karlinsky, Joseph Shtok, Sivan Harary, Eli Schwartz, Amit Aides, Rogério Schmidt Feris, Raja Giryes, and Alexander M. Bronstein. Repmet: Representative-based metric learning for classification and few-shot object detection. In *CVPR 2019*, pages 5197–5206. Computer Vision Foundation / IEEE, 2019. 1

[13] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML 2015*. Lille, 2015. 2

[14] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009. 4

[15] Piotr Koniusz, Lei Wang, and Anoop Cherian. Tensor representations for action recognition. *TPAMI*, 2020. 2

[16] Piotr Koniusz, Fei Yan, Philippe-Henri Gosselin, and Krystian Mikolajczyk. Higher-order occurrence pooling for bags-of-words: Visual concept detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(2):313–326, 2017. 2

[17] Piotr Koniusz, Fei Yan, Philippe-Henri Gosselin, and Krystian Mikolajczyk. Higher-order occurrence pooling on mid- and low-level features: Visual concept detection. *Tech. Report*, 2013. 2

[18] Piotr Koniusz and Hongguang Zhang. Power normalizations in fine-grained image, few-shot image and graph classification. *TPAMI*, 2020. 2, 3

[19] Piotr Koniusz, Hongguang Zhang, and Fatih Porikli. A deeper look at power normalizations. In *CVPR 2018*, pages 5774–5783. IEEE Computer Society, 2018. 2

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, 2017. 2

[21] Aoxue Li and Zhenguo Li. Transformation invariant few-shot object detection. In *CVPR 2021*, pages 3094–3102, 2021. 6

[22] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *CVPR 2019*, pages 510–519. IEEE, 2019. 2

[23] Yiting Li, Haiyue Zhu, Yu Cheng, Wenxin Wang, Chek Sing Teo, Cheng Xiang, Prahlad Vadakkepat, and Tong Heng Lee. Few-shot object detection via classification refinement and distractor retreatment. In *CVPR 2021*, pages 15395–15403, 2021. 6

[24] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *ECCV 2014*, pages 740–755. Springer, 2014. 6, 13

[25] Songtao Liu, Di Huang, and Yunhong Wang. Receptive field block net for accurate and fast object detection. In *ECCV 2018*, pages 404–419. Springer, 2018. 2

[26] 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. *CoRR*, abs/2103.14030, 2021. 13

[27] Changsheng Lu and Piotr Koniusz. Few-shot keypoint detection with uncertainty learning for unseen species. In *CVPR*, 2022. 2

[28] Saimunur Rahman, Lei Wang, Changming Sun, and Luping Zhou. Redro: Efficiently learning large-sized spd visual representation. In *ECCV 2020*, 2020. 2

[29] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6517–6525. IEEE Computer Society, 2017. 1, 2

[30] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. 1, 2

[31] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS 2015*, pages 91–99. IEEE Computer Society, 2015. 1, 2, 6

[32] Ya-Fang Shih, Yang-Ming Yeh, Yen-Yu Lin, Ming-Fang Weng, Yi-Chang Lu, and Yung-Yu Chuang. Deep co-occurrence feature learning for visual object recognition. In *CVPR 2017*, pages 7302–7311. IEEE Computer Society, 2017. 2

[33] Pranav Shyam, Shubham Gupta, and Ambedkar Dukkipati. Attentive recurrent comparators. In *ICML 2017*, pages 3173–3181. PMLR, 2017. 2

[34] Christian Simon, Piotr Koniusz, and Mehrtash Harandi. Meta-learning for multi-label few-shot classification. In *WACV*, pages 3951–3960, 2022. 2

[35] Christian Simon, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. Adaptive subspaces for few-shot learning. In *CVPR*, 2020. 2

[36] Christian Simon, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. On modulating the gradient for meta-learning. In *ECCV*, 2020. 2

[37] Bharat Singh, Mahyar Najibi, and Larry S. Davis. SNIPER: efficient multi-scale training. In *NIPS 2018*, pages 9333–9343, 2018. 2

[38] Alexander J. Smola and Risi Kondor. Kernels and regularization on graphs, 2003. 1, 2

[39] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *NIPS 2017*, pages 4077–4087, 2017. 1, 2

[40] Bo Sun, Banghuai Li, Shengcai Cai, Ye Yuan, and Chi Zhang. FSCE: few-shot object detection via contrastive proposal encoding. *CoRR*, abs/2103.05950, 2021. 6

[41] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR 2018*, pages 1199–1208. IEEE Computer Society, 2018. 1, 2

[42] Oncel Tuzel, Fatih Porikli, and Peter Meer. Region covariance: A fast descriptor for detection and classification. In Ales Leonardis, Horst Bischof, and Axel Pinz, editors, *ECCV 2006*, pages 589–600. Springer, 2006. 2

[43] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NIPS 2016*, pages 3630–3638, 2016. 1

[44] Paul A. Viola and Michael J. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR 2001*, pages 511–518. IEEE Computer Society, 2001. 4

[45] Xin Wang, Thomas E. Huang, Joseph Gonzalez, Trevor Darrell, and Fisher Yu. Frustratingly simple few-shot object detection. In *ICML 2020*, pages 9919–9928. PMLR, 2020. 2, 6

[46] Yulin Wang, Xuran Pan, Shiji Song, Hong Zhang, Gao Huang, and Cheng Wu. Implicit semantic data augmentation for deep networks. *NIPS*, 32:12635–12644, 2019. 5, 11

[47] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *ICML*, pages 1113–1120, 2009. 1, 3, 5

[48] Aming Wu, Yahong Han, Linchao Zhu, and Yi Yang. Universal-prototype enhancing for few-shot object detection. In *ICCV*, pages 9567–9576, October 2021. 6

[49] Jiaxi Wu, Songtao Liu, Di Huang, and Yunhong Wang. Multi-scale positive sample refinement for few-shot object detection. In *ECCV 2020*, pages 456–472. Springer, 2020. 2, 6

[50] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR 2017*, pages 5987–5995. IEEE Computer Society, 2017. 2

[51] Andrés Romero Mier y Terán, Michèle Gouiffès, and Lionel Lacassagne. Enhanced local binary covariance matrices (ELBCM) for texture analysis and object tracking. In *MIRAGE*, pages 10:1–10:8. ACM, 2013. 2

[52] Xiaopeng Yan, Ziliang Chen, Anni Xu, Xiaoxi Wang, Xiaodan Liang, and Liang Lin. Meta R-CNN: towards general solver for instance-level low-shot learning. In *ICCV 2019*, pages 9576–9585. IEEE, 2019. 1, 2, 6

[53] Yukuan Yang, Fangyun Wei, Miaojing Shi, and Guoqi Li. Restoring negative information in few-shot object detection. In *NIPS*, 2020. 2, 6

[54] Hongguang Zhang and Piotr Koniusz. Power normalizing second-order similarity network for few-shot learning. In *WACV 2019*, pages 1185–1193. IEEE, 2019. 2

[55] Hongguang Zhang, Hongdong Li, and Piotr Koniusz. Multi-level second-order few-shot learning. *IEEE Transactions on Multimedia*, 2022. 2

[56] Hongguang Zhang, Li Zhang, Xiaojuan Qi, Hongdong Li, Philip H. S. Torr, and Piotr Koniusz. Few-shot action recognition with permutation-invariant attention. In *ECCV*, 2020. 2

[57] Jianjia Zhang, Lei Wang, Luping Zhou, and Wanqing Li. Beyond covariance: SICE and kernel based visual feature representation. *Int. J. Comput. Vis.*, 129(2):300–320, 2021. 1

[58] Shan Zhang, Dawei Luo, Lei Wang, and Piotr Koniusz. Few-shot object detection by second-order pooling. In *ACCV*, 2020. 1, 2, 6, 8

[59] Minghang Zheng, Peng Gao, Xiaogang Wang, Hongsheng Li, and Hao Dong. End-to-end object detection with adaptive clustering transformer. *CoRR*, abs/2011.09315, 2020. 2

[60] Hao Zhu and Piotr Koniusz. Ease: Unsupervised discriminant subspace learning for transductive few-shot learnin. In *CVPR*, 2022. 2

[61] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: deformable transformers for end-to-end object detection. In *ICLR 2021*. OpenReview.net, 2021. 2