Abstract

Few-shot learning has attracted significant scientific interest in the past decade due to its applicability to visual tasks with a natural long-tailed distribution such as object detection. This paper introduces a novel and flexible few-shot object detection approach which can be adapted effortlessly to any candidate-based object detection framework. In particular, our proposed kFEW component leverages a kNN retrieval technique over the regions of interest space to build both a class-distribution and a weighted aggregated embedding conditioned by the recovered neighbours. The obtained kNN feature representation is used to drive the training process without any additional trainable parameters as well as during inference time by steering the assumed confidence and the predicted box coordinates of the detection model.

We perform extensive experiments and ablation studies on MS COCO and Pascal VOC proving its efficiency and state-of-the-art results (by a margin of 2.3 mAP points on MS COCO and by a margin of 2.5 mAP points on Pascal VOC) in the context of few-shot-object detection. Additionally, we demonstrate its versatility and ease-of-integration aspect by incorporating over competitive few-shot object detection methods and providing superior results.

1. Introduction

In the past decade, significant advances have been made in computer vision on challenging topics such as recognition [9, 14], reconstruction [30] and context reasoning [19]. Enhanced GPU capabilities and the availability of large scale datasets enabled the development of robust and generalized methods [15, 27, 11, 5, 22, 4]. The collection of large scale datasets usually entails laborious processes involving intense human effort or sensory equipment for measurement-based annotations (e.g. complex motion capture systems for 3D pose annotations). However, not all problems are scalable or approachable through such data acquisition mechanics. Certain use cases are characterized by sparse information representations where data is difficult to obtain even with abundant resources. Object detection is a representative use case, where the natural distribution of searched objects is usually long-tailed in the sense that certain classes are densely represented while others are heavily underrepresented. This data sparsity issue motivated the emergence of the few-shot object detection (FSOD) paradigm.

The FSOD setup aims to detect unseen (novel) object categories using very few training samples (i.e. ≤ 30). One of the most relevant FSOD learning strategies involves a two stage fine-tuning approach [33, 20]. The first stage consists of training an object detector on a large corpus with base object categories that are densely represented. As a result of the learning process, the feature representation retrieved from the model should accommodate in a generic fashion the patterns and visual appearance encoded within the train data for the base categories. The second stage consists of adapting the trained detector to novel object categories that are heavily underrepresented. This task involves a collection of challenges such as (i) covariance shift between the distributions of novel and base classes and (ii) high ambiguity between the visual representation of novel and base classes requiring constraining mechanisms to make the embeddings of novel classes discriminative.
Figure 2. **Detailed overview of the proposed kFEW method.** For each proposal \( r \) we obtain a list of neighbours \( Q^{kNN} \) which is used for defining \( P_{kFEW} \) and \( A_{kFEW} \), the conditional class probability and the encoded feature representations, respectively. Having no learnable parameters and depending only on the \( Q \) feature, our \( kFEW \) can be used as an additional optimization strategy during train time (red flow) and at inference time (dotted flow). While any candidate-based detection model can be used, for illustration purposes the Faster-RCNN [36] framework was employed.

We observe that a clear area to strengthen the typical FSOD approach is to focus on exploiting the entire provided few-shot input space. Traditionally, the *novel* training samples are analysed one batch at a time, without taking a glimpse at the entire data context. To overcome this bottleneck, we propose an innovative plug-and-play strategy called \( kFEW \) (*k*NN over FEW-shot learning) that improves FSOD by incorporating a conditional aggregated representation of the entire input space during inference and training. Our algorithm leverages a *k*-Nearest Neighbour (*k*-NN) feature weighting technique [43] operating on region proposal embeddings which are fed afterwards to the region of interest classifier and regressor, respectively. The motivation behind the usage of the *k*-NN embeddings is to leverage the fact that the few-shot data setup allows for retrieving on-the-fly a conditional embedding and a class probability for the analysed image proposals. *k*-NN is an ideal fit for few-shot data setup, as the neighbour search space is small enough for the best computational efficiency. To the best of our knowledge, we are the first to propose such a FSOD mechanism which incorporates a *k*-NN embedding strategy over the space defined by the training data.

Any candidate-based object detection framework can be utilized with our proposed \( kFEW \) as we demonstrate in the experimental section with the methods DeFRCN [33] and TFA [42]. However, in the methodology section, we use the Faster-RCNN [36] to clearly illustrate and explain our proposed methodology. \( kFEW \) does not depend on any learnable parameters (*i.e.* the total number of trainable parameters of the model remains constant) and it evidences *state-of-the-art* performance in the context of generalized (see figure [1]) and standard few-shot-object detection on Microsoft COCO [27] and Pascal VOC [6] datasets concerning \( K = 1, 5, 10, 30 \) shot learning setups. Additionally, it can be easily incorporated within any 2-stage FSOD framework. The impact is demonstrated in a 2-fold manner: (1) during train time operating as feature-based constraint for the object proposal classifier / regressor and (2) during inference time using linear interpolation between the conditional class probabilities from the *k*-NN retrieved space and the object proposal classifier. Moreover, the aggregated feature representation of the retrieved neighbours can be added as a weighted factor to the encoding of the object proposal regressor.

2. Related Work

Object detection [36, 25, 9, 1, 26, 35, 28, 2, 39] is one of the classical computer vision problems. It implies simultaneous classification and localization of object instances from the searched image space. Over the years, two branches of methodologies were developed, (*i*) single-pass methods [28, 35, 34, 39] aiming at recovering the object instances directly via dense geometric or appearance pattern search and (*ii*) double-pass methods [24, 2, 36, 25, 9] which leverage a class-agnostic candidate region proposal extraction phase of potential searched object instances.

One of the most famous lines of work concerning the double-phase type of detectors is the Region-Based Convolutional Neural Network (RCNN) [36, 13, 8, 25, 9]. The central idea behind this line of work is to have a convolutional neural network structure which efficiently en-
codes the image information in a generic manner. Next, the encoded information is transformed into a list of class-agnostic candidate objects using a region proposal network (RPN). Lastly, the objects from the proposal list are pooled to region-of-interest (ROI) features and a refined list of objects defined by class labels and bounding box coordinates is obtained using a classifier and a regressor head.

2.1. Few-Shot Learning

Few short learning [21, 17, 40, 43, 38] has been the focus of increased scientific attention recently tackling data availability challenges and learning tasks requiring reduced samples sizes or sparse data representations.

Main avenues of research focused on meta-learning [37] and metric-learning [38, 41]. We are interested in the later, which investigates the task of learning an embedding function which can be used to project the input samples to a space which enhances the similarity between same class labels and the dissimilarity between different class labels.

Although it initially became obsolete in the deep learning era, the k-NN technique resurfaced in sub-domains such as few-shot learning. In this work, we are interested in the property of k-NN to build data associations in various feature space representations which proved to be of pivotal role especially in the context of underrepresented data distributions. One such example is the work of [16] tackling the task of image classification where a feature store representation is being leveraged as a penalty term to the final classification layer via the cross-entropy loss function by using the conditional probability of the retrieved neighbours.

2.2. Few-Shot Object Detection

Traditional object detection frameworks are trained using stochastic gradient descent learning strategies operating under the assumption that the train data is uniformly distributed. However, in practical long-tail distribution setups this assumption does not hold. Additionally, when multiple retraining rounds are involved, the phenomena of catastrophic forgetting [7, 10] might occur. To avoid this problem, recent approaches on the task of FSOD [33, 42, 48, 20, 51, 23, 50, 12] use a two-stage fine tuning approach. In the first stage, a base detector is pre-trained on a large set containing base object categories, while in the second stage, a fine-tuning train step is involved with different feature embedding heuristics to prevent catastrophic forgetting for base classes and to easily on-board novel classes.

Similar to our approach, the authors of [20] propose a pseudo-labelling framework based on Faster-RCNN [45] which leverages a k-NN strategy resonating with an active learning strategy. Firstly, using a baseline few-shot detector they obtain a raw pool of initial candidates over an unlabelled set of images. Secondly, the raw pool is filtered using a k-NN classifier constructed using the same few-shot image set. Lastly, the filtered set is reused after processing via a box corrector to better refine the initial few-shot detector. Different from their work, our proposed kFEW leverages the few-shot train set with a k-NN feature constraining strategy inspired from [16] to better model the embedding space of the region-of-interest. Moreover, we incorporate in an end-to-end fashion the k-NN inside the training / inference pipelines.

3. Methodology

In the following we present our proposed plug-and-play kFEW framework emphasizing the key innovative aspects.

Let there be an image \( I \in \mathbb{R}^{w \times h \times 3} \). The objective is to retrieve a list \( \mathcal{Y} = \{y_i\}_{i=1}^{N} \) of object proposals where \( y_i = (c_i, b_i) \) with \( b_i \in [0, 1]^4 \) representing the bounding box coordinates of the proposal with respect to image space and \( c_i \in \mathcal{C}_{\text{all}} \) representing the target class, where \( \mathcal{C}_{\text{all}} = \mathcal{C}_{\text{base}} \cup \mathcal{C}_{\text{novel}} \). Class categories \( \mathcal{C}_{\text{base}} \) and \( \mathcal{C}_{\text{novel}} \) correspond to base classes which are heavily represented in the training set and novel classes, respectively, which are represented by the few-shot data support. Moreover, both class categories are non-overlapping, \( \mathcal{C}_{\text{base}} \cap \mathcal{C}_{\text{novel}} = \emptyset \). A major prerequisite of FSOD is to have either a pretrained host detector or a dataset with dense representations for the \( \mathcal{C}_{\text{base}} \) categories.

In the general case, the candidate-based detection framework can be summarized with the following computational pipeline:

\[
\Psi_{\text{OBJ}}(\cdot) = \Psi_{\text{CLS}} \circ \Psi_{\text{ROI}} \circ \Psi_{\text{RPN}} \circ \Psi_{\text{ENC}}(\cdot) \quad (1)
\]
\[
\Psi_{\text{OBJ}}(\cdot) = \Psi_{\text{BBX}} \circ \Psi_{\text{ROI}} \circ \Psi_{\text{RPN}} \circ \Psi_{\text{ENC}}(\cdot) \quad (2)
\]

where \( \Psi_{\text{ENC}} \) is a common image encoding backbone (usually either a ResNet50, ResNet101 [13] or SWIN [29] backbone). \( \Psi_{\text{RPN}} \) is a class-agnostic RPN, \( \Psi_{\text{ROI}} \) is a ROI feature encoder which link the proposals from \( \Psi_{\text{RPN}} \) with the image encoding provided by \( \Psi_{\text{ENC}} \). \( \Psi_{\text{CLS}} \) and \( \Psi_{\text{BBX}} \) represent classification and regression heads, respectively, which align the proposed regions of interest to a set of object classes and bounding boxes, respectively. Following the same representation, we use \( \mathcal{C} \) and \( \mathcal{B} \) to denote the object classes and bounding boxes, respectively in the left hand side of equations [1] and [2]. The parameters of \( \Psi_{\text{ENC}}, \Psi_{\text{RPN}} \) and \( \Psi_{\text{ROI}} \) are shared for both \( \Psi_{\text{OBJ}} \) and \( \Psi_{\text{OBJ}} \). Additionally, the entire ensemble is trained end-to-end with losses penalizing the RPN (for better background separation of the proposals and anchor refinement) and the classification / regression heads:

\[
\mathcal{L} = \mathcal{L}_{\text{OBJ}} + \mathcal{L}_{\text{OBJ}} + \mathcal{L}_{\text{RPN}} \quad (3)
\]
Thus, for the input image \( I \), the model outputs a set of predictions \( \hat{y} = (\Psi_{OBJc}(I), \Psi_{OBJb}(I)) \). For the two-phase FSOD use-case, first, the entire ensemble is trained using \( C_{base} \) data followed by a fine tuning step using \( kFEW \) leveraging \( C_{novel} \) data.

The \( k \)-NN mechanism behind our proposed \( kFEW \) component requires a set of object instances \( D = \{d_1 \ldots d_M\} \), where \( d_i \in \mathbb{R}^w \times h \times 3 \), with attached class labels \( \{c_1 \ldots c_M\} \) where \( c_i \in \mathcal{C}_{all} \). They are used inside \( kFEW \) via their \( \Psi_{ROI} \) feature encoder representation. For a better understanding of the proposed methodology, we consider \( \Psi_{ROI}(D) = \{\Psi_{ROI}(d_1) \ldots \Psi_{ROI}(d_M)\} \) represents the feature store where \( k \)-NN operates, where term \( M \) represents the cardinality of our feature store. For simplicity, we will use \( Q = \{q_1 \ldots q_M\} \).

The algorithmic steps of the proposed \( kFEW \) can be decomposed as follows.

**STEP I: Retrieve \( k \) nearest object instances.** Given a class-agnostic region proposal \( r \in \Psi_{EP}(I) \) obtained as a result of the RPN from \( I \), we compute the Euclidean distance between \( \Psi_{ROI}(r) \) and every element from \( Q \). To simplify notation, we denote \( r = \Psi_{ROI}(r) \). Thus, we obtain the set \( \{\|q - r\|_2 \mid q \in Q\} \). Prior to applying the \( \delta \) distance function, the descriptors are normalized using their \( l^2 \) norm and the aggregated mean from set \( Q \). As a result of the \( k \)-NN process, we obtain a list \( Q^{\text{NN}} \) of neighbours, where \( |Q^{\text{NN}}| = k \), with \( k \) representing the number of retrieved neighbours. This list is implicitly split in sublists \( Q^{\text{NN}}_{c_i} \), where \( c_i \in \mathcal{C}_{all} \), according to their class label.

**STEP II: Encode the retrieved neighbour information.** Next, we build a probability distribution using the classes of the retrieved neighbours list \( Q^{\text{NN}} \). Intuitively, we plan to derive probability distribution of the \( k \)-NN space conditioned by the set of class labels \( \mathcal{C}_{all} \). Thus, we obtain the following conditional class distribution given region proposal \( r \)

\[
\mathcal{C}_{all}|r \sim \left( \begin{array}{cccc}
  c_1 & c_2 & c_3 & \cdots & c_{|\mathcal{C}_{all}|} \\
  p(c_1|r) & p(c_2|r) & p(c_3|r) & \cdots & p(c_{|\mathcal{C}_{all}|}|r)
\end{array} \right)
\]

where probability \( p(\cdot) \) is defined as follows, with \( q, r \in \mathbb{R}^d \)

\[
p(c_i|r) = \frac{\sum_{q \in Q^{\text{NN}}_{c_i}} \exp(-\|q - r\|_2/\tau)}{\sum_{c_j \in \mathcal{C}_{all}} \sum_{q \in Q^{\text{NN}}_{c_j}} \exp(-\|q - r\|_2/\tau)} \tag{4}
\]

The spread of the exponential factor \( \tau \) inside the probability is determined using validation. A higher value of \( \tau \) produce a more flatten probability distribution. As a result of this operation, we consider the array of resulted class probabilities \( P_{\text{FWEK}} = [p(c_1|\tau), p(c_2|\tau) \cdots p(c_{|\mathcal{C}_{all}|}|\tau)] \).

Additionally, we compute the weighted average of all the encoded feature representations for the retrieved neighbours within \( Q^{\text{NN}} \)

\[
A_{\text{FWEK}} = \sum_{q \in Q^{\text{NN}}} \exp(-\|q - r\|_2/\tau) \cdot q \sum_{q \in Q^{\text{NN}}} \exp(-\|q - r\|_2/\tau) \tag{5}
\]

**STEP III: Incorporate the \( k \)-NN encoding inside the FSOD pipeline.** Lastly, we constrain the FSOD pipeline using the previously computed information, \( P_{\text{FWEK}} \) and \( A_{\text{FWEK}} \). The term \( P_{\text{FWEK}} \) is used as weighting factor via the negative log-likelihood loss \([49]\) for the \( \mathcal{L}_{\text{OBJ}c} \) loss component. Intuitively, we aim at penalizing the class information of the object proposals using the \( k \)-NN encoded information. Thus, the classification branch which accommodates the \( kFEW \) strategy is penalized by the following weighted classification loss:

\[
\mathcal{L}_{\text{OBJ}c}^{\text{FWEK}} = (1 + \beta \cdot \mathcal{L}_{\text{HLL}}(P_{\text{FWEK}})) \cdot \mathcal{L}_{\text{OBJ}c} \tag{6}
\]

where \( \beta \in [0,1] \) represents a scaling factor to determine the impact of the retrieved \( k \)-NN conditional distribution and is determined by validation. The term \( \mathcal{L}_{\text{OBJ}c}^{\text{FWEK}} \) replaces the \( \mathcal{L}_{\text{OBJ}c} \) term from equation\([3]\).

Furthermore, the encoded feature representation \( A_{\text{FWEK}} \) is used as a weighted factor when passed to the \( \Psi_{BBX} \) head and thus, the regression branch influenced by \( kFEW \) becomes:

\[
\Psi_{\text{OBJ}c}^{kFEW}(\cdot) = \Psi_{\text{BBX}} \circ ((1 - \lambda)A_{\text{FWEK}} + \lambda \Psi_{\text{ROI}}) \circ \Psi_{\text{RPN}} \circ \Psi_{\text{ENC}}(\cdot)
\]

where parameter \( \lambda \in [0,1] \) is a linear interpolation term which weights the balance between original feature representation and the \( kFEW \) feature representation.

Another major advantage of our proposed framework is that it can be easily incorporated directly during inference time by influencing the predicted class probabilities using the conditional probability distribution derived from the retrieved \( k \)-NN space. Thus, using the resulted conditional probability \( P_{\text{FWEK}} \) we can derive the following \( kFEW \) influenced classification branch:

\[
\Psi_{\text{OBJ}c}^{kFEW} = (1 - \alpha)P_{\text{FWEK}} + \alpha P_{\text{OBJ}c} \tag{7}
\]
Deep learning framework. We use a SGD optimizer with V100 GPU at 16 GB of GPU RAM and the Pytorch [31] deep learning framework. We use a SGD optimizer with weight decay of $10^{-5}$ with a batch size of 16 to train our models. The number of training epochs varies mostly on the dataset used and the number of shots available during train time.

4. Experiments

We experiment on the intensely explored datasets Pascal VOC [6] and MSCOCO [27] proving the capabilities of kFEW. Our framework does not operate in a standalone object detection mode; instead, it can be used as an extension to any two-stage detection framework. As such, we integrate it with 2 strong baselines, TFA [42] and DeFRCN [33], proving superior results over their original configuration and state-of-the-art results. The evaluation is performed on the standard FSOD use-case which involves few-shot learning and evaluation on the novel classes only and the generalized few-shot object detection (GFSOD) setup which implies training and evaluation in the few-shot context for both categories, novel and base. The latter is a more challenging and practical scenario as it necessitates top performance for novel classes while improving the performance of the base classes. For our experimental setup, we use a system with an Intel Xeon QuadCore 2.5 GHz architecture, with 32 GB system RAM and 4 NVIDIA Tesla V100 GPU at 16 GB of GPU RAM and the Pytorch [31] deep learning framework. We use a SGD optimizer with

Table 1. Experimental results on Pascal VOC. We report AP results at IoU threshold of 0.5 for both evaluation setups FSOD and GFSOD reporting superior performance with respect to the host methods as well as demonstrating state-of-the-art results on the majority of the few-shot scenarios. State-of-the-art results are denoted with red text and second best results are marked with blue text.

<table>
<thead>
<tr>
<th>Method</th>
<th>GFSOD</th>
<th>Novel Set 1</th>
<th>Novel Set 2</th>
<th>Novel Set 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>YOLO-ft [18]</td>
<td>✓</td>
<td>6.6</td>
<td>24.8</td>
<td>38.6</td>
</tr>
<tr>
<td>FRCN-ft [47]</td>
<td>✓</td>
<td>13.8</td>
<td>41.5</td>
<td>45.6</td>
</tr>
<tr>
<td>LSTD [3]</td>
<td>✓</td>
<td>8.2</td>
<td>29.1</td>
<td>38.5</td>
</tr>
<tr>
<td>MetaDet [44]</td>
<td>✓</td>
<td>14.8</td>
<td>33.9</td>
<td>47.2</td>
</tr>
<tr>
<td>Meta R-CNN [47]</td>
<td>✓</td>
<td>18.9</td>
<td>36.8</td>
<td>49.6</td>
</tr>
<tr>
<td>TFA [32]</td>
<td>✓</td>
<td>19.9</td>
<td>45.7</td>
<td>51.5</td>
</tr>
<tr>
<td>TFA w/kFEW</td>
<td>✓</td>
<td>31.1</td>
<td>55.7</td>
<td>37.8</td>
</tr>
<tr>
<td>TFA w/kFEW</td>
<td>✓</td>
<td>32.2</td>
<td>58.6</td>
<td>40.0</td>
</tr>
<tr>
<td>MCSR [45]</td>
<td>✓</td>
<td>41.7</td>
<td>55.2</td>
<td>61.8</td>
</tr>
<tr>
<td>DeFRCN [33]</td>
<td>✓</td>
<td>53.6</td>
<td>62.2</td>
<td>59.3</td>
</tr>
<tr>
<td>DeFRCN w/kFEW</td>
<td>✓</td>
<td>54.7</td>
<td>63.4</td>
<td>60.5</td>
</tr>
<tr>
<td>FRCN-ft [47]</td>
<td>✓</td>
<td>9.9</td>
<td>28.0</td>
<td>52.0</td>
</tr>
<tr>
<td>FSRW [18]</td>
<td>✓</td>
<td>14.2</td>
<td>36.5</td>
<td>35.6</td>
</tr>
<tr>
<td>TFA [32]</td>
<td>✓</td>
<td>25.3</td>
<td>47.9</td>
<td>52.8</td>
</tr>
<tr>
<td>TFA w/kFEW</td>
<td>✓</td>
<td>26.1</td>
<td>49.7</td>
<td>54.1</td>
</tr>
<tr>
<td>FSDetView [46]</td>
<td>✓</td>
<td>24.2</td>
<td>49.1</td>
<td>57.4</td>
</tr>
<tr>
<td>DeFRCN [33]</td>
<td>✓</td>
<td>40.2</td>
<td>65.0</td>
<td>66.6</td>
</tr>
<tr>
<td>DeFRCN w/kFEW</td>
<td>✓</td>
<td>42.7</td>
<td>65.3</td>
<td>66.8</td>
</tr>
</tbody>
</table>

where $\alpha \in [0, 1]$ is a linear interpolation term and $P_{CB, C}$ is the output of $\Psi_{CB, C}$. The same inference heuristic is applied with $k_{FWE}$. An overview of our proposed pipeline is displayed in figure 2. We illustrate separately the train and inference flow and as mentioned, our $k_{FEW}$ is not bounded by any means to Faster-RCNN and can be immediately incorporated within any learning framework with the added benefit of no additional learnable parameters.

4.1. Pascal VOC

This dataset contains depictions of 20 generic object classes. The experimental setup for FSOD implies a split of the overall categories for base and novel into 15 and 5 categories, respectively. We follow the protocol from [20, 33, 32] of having 3 random split groups of novel and base for the few-shot scenarios of $K = 1, 5, 10$.

Our proposed $k_{FEW}$ is incorporated in both TFA and DeFRCN frameworks during few-shot fine-tuning stage. Evaluation metrics are available in table 1. The metric we use is the average precision for novel classes computed at an intersection over union (IoU) threshold of 0.5. The evaluation is performed as an average over 30 repeated runs for each configuration. Our method is able to achieve consistent superior performance over existing methods by a significant margin on different splits and multiple shot scenarios. These improvements apply to both FSOD and GFSOD evaluation setups.

4.2. Microsoft COCO

We follow the same evaluation setup introduced in [18] [32]. We use the common 20 object categories with Pascal VOC as novel and the rest of 60 as base categories. To be in line with previous works [33, 42, 20], the metric we report is the mean AP obtained for all the IoUs between 0.5 and
Table 2. (a) Performance comparison for different update frequencies of Q. Intuitively, Q should encode the current model feature representation. Thus, it is natural to have the best performance with a frequency update performed every epoch. (b) Performance impact of feature store sizes for DeFRCN with kFEW on 1-shot learning setup. As the feature store Q size is increased, the retrieved k-NN representations for the region proposals are better defined which leads to an increased evaluation score. (c) Impact of PFEW at inference time. We vary the α interpolation term inside equation 7. When varying the linear interpolation α inside equation 7 the highest impact is observed for α = 0.4. (d) Impact of AFEW and PFEW w.r.t. inference / training routine. Performance results with DeFRCN model using k-FEW on the 1-shot setup for novel classes leveraging differently the weighted feature representation AFEW and the conditional probability PFEW.

<table>
<thead>
<tr>
<th>Update Frequency</th>
<th>APn</th>
<th>Size of Q</th>
<th>APk</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>17.3</td>
<td>1-shot</td>
<td>7.95</td>
</tr>
<tr>
<td>INIT</td>
<td>17.7</td>
<td>5-shot</td>
<td>8.17</td>
</tr>
<tr>
<td>1 EPOCH</td>
<td>18.0</td>
<td>10-shot</td>
<td>8.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30-shot</td>
<td>9.21</td>
</tr>
</tbody>
</table>

Table 3. We analyse the performance gap by freezing different parts of DeFRCN during training on COCO. The performance for the base classes is highest when the ENC / RPN are frozen. On the other hand, the top performance for novel classes is achieved with all the modules unlocked.

<table>
<thead>
<tr>
<th>FEW</th>
<th>FEW</th>
<th>FEW /FEW</th>
<th>APp</th>
<th>APk</th>
<th>APn</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>31.6</td>
<td>36.6</td>
<td>16.2</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>30.8</td>
<td>35.1</td>
<td>17.8</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>31.4</td>
<td>36.5</td>
<td>16.1</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>30.7</td>
<td>35.0</td>
<td>17.9</td>
</tr>
</tbody>
</table>

Are there differences between PFEW and AFEW? The intuition is to boost the performance of both ΨOBJ and ΨOBJ given that they optimize different aspects of the detection task, the predicted object’s class and its image localization. Each term acts as an individual weight contributor, one with the conditional class probability and the other with a data inferred embedding. In table 2(d) we study the impact of each term during inference and train usage (7.95 AP on the novel classes). As expected, the highest performance is achieved using both, during both training and inference time. A non-trivial gain is obtained even if both terms are used during inference only.

What is the computational overhead brought by kFEW? In our default setup, we use the few-shot training set as Q to have a fair evaluation comparison against the other baselines. However, in practice, the feature store can be different and as proven in table 2(b), increased performance is associated with larger feature stores (1.26 gain in terms of APn). By increasing the size of Q, both PFEW and AFEW encode a better representation for the queried region proposal as the search space is better defined. Moreover, in table 5 we study the computational time impact of using kFEW on the MS COCO dataset for each DeFRCN [33] and TFA [42]. Notice that the added time is insignificant (17.6 ms for DeFRCN and 11.7 ms for TFA) compared to the performance gain.

Why not rely predictions on kFEW only? One might think, why not use only the probabilities and the encodings obtained via kFEW, without reintroducing them in the inference framework. We analysed the impact of such a decision in table 2(c). By using only kFEW (i.e. α = 0) the overall performance decreases. The best AP is obtained for
Figure 3. **Retrieved neighbours from** $Q$ **together with top 16 probability class scores.** From left to right we have the top 16 probability scores from $P_{FEW}$, the predicted bounding box, its corresponding class and the 10 nearest neighbours from the feature store $Q$. Notice that for each sample, the highest probability score corresponds to the target class, and the following ones correspond either to similar objects in terms of semantic meaning or to objects which tend to appear in a similar context.
an interpolation factor of 0.4, proving complementary performance gain between the local processing pipeline and our data-dependent flow.

**How often should $Q$ be updated?** A key ingredient of our framework is represented by the feature store $Q$ of object proposals. As described in methodology, these are obtained using the $\Psi_{SOD}$ embedding function. However, this representation is not static throughout the training process, and it should constantly change. Ideally, $Q$ should be updated every iteration; however, due to computational costs, it will be very costly and inefficient. In table 2(a) we analyse the performance impact of the update frequency for the feature store representation: (I) without $kFEW$, (II) using pre-trained model from base classes and (III) every epoch. This experiment was performed using the DeFRCN framework as host on the 10-shot MSCOCO setup. We report the mean average precision on the IoU interval $[0.5, 1]$. By updating $Q$ each epoch, we have a gain of 0.3 $mAP$ points compared to the setup where $Q$ is obtained using the $\Psi_{SOD}$ from base classes without any update during train time.

**Which component influences more the overall performance via $kFEW$?** In table 3 we analyse the impact of $kFEW$ with different subcomponents of DeFRCN frozen during training. As expected, the performance for base classes is highest when both ENC and RPN are frozen (i.e. $36.6AP^b$), as a direct consequence of the pre-training. On the other side, the performance for novel classes peaks with everything unblocked as it is easier to accommodate the novel classes (i.e. 17.9$AP^b$).

**Can $kFEW$ improve the base detector?** The $kFEW$ mechanism should improve the base training as well, however in this scenario, $Q$ will be the entire base training set, making it unfeasible to be updated on a regular basis due to the large computational overhead. In practice, it becomes unsuitable as it will require heavy optimisation such as k-NN search based on local sensitive hashing.

### 5. Conclusions

In this paper we introduced $kFEW$, a k-NN feature retrieval-based framework for few-shot object detection which can be successfully incorporated during training as well as inference routines within any proposal-based object framework without the necessity of any additional learnable parameters. Moreover, we successfully demonstrate the versatility aspect of $kFEW$ by incorporating it with 2 strong few-shot object detection frameworks, TFA [42] and DeFRCN [33] while proving state-of-the-art results in the context of $K = 1, 5, 10, 30$ shot learning scenarios on the challenging Microsoft COCO and Pascal VOC benchmarks for FSOD / GFSOD. In addition, we perform an extensive list of ablation studies consolidating the intuition behind our $kFEW$ framework and the extent to which it can be utilized.
References


