A Fusion Spatial Attention Approach for Few-shot Learning

Heda Song[a], Bowen Deng[b], Michael Pound[b], Ender Özcan[a], Isaac Triguero[a]

[a] *Computational Optimisation and Learning (COL) Lab. School of Computer Science. University of Nottingham. Jubilee Campus. Nottingham NG8 1BB, United Kingdom*
[b] *Computer Vision Laboratory, School of Computer Science, University of Nottingham, Jubilee Campus. Nottingham NG8 1BB, United Kingdom*

Abstract

Few-shot learning is a challenging problem in computer vision that aims to learn a new visual concept from very limited data. A core issue is that there is a large amount of uncertainty introduced by the small training set. For example, the few images may include cluttered backgrounds or different scales of objects. Existing approaches mostly address this problem from either the original image space or the embedding space by using meta-learning. To the best of our knowledge, none of them tackle this problem from both spaces jointly. To this end, we propose a fusion spatial attention approach that performs spatial attention in both image and embedding spaces. In the image space, we employ a Saliency Object Detection (SOD) module to extract the saliency map of an image and provide it to the network as an additional channel. In the embedding space, we propose an Adaptive Pooling (Ada-P) module tailored to few-shot learning that introduces a meta-learner to adaptively fuse local features of the feature maps for each individual embedding. The fusion process assigns different pooling weights to the features at different spatial locations. Then, weighted pooling can be conducted over an embedding to fuse local information, which can avoid losing useful information by considering the spatial importance of the features. The SOD and Ada-P modules can be used within a plug-and-play module and incorporated into various existing few-shot learning approaches. We empirically

Email addresses: heda.song@nottingham.ac.uk (Heda Song), bowen.deng@nottingham.ac.uk (Bowen Deng), michael.pound@nottingham.ac.uk (Michael Pound), ender.ozcan@nottingham.ac.uk (Ender Özcan), isaac.triguero@nottingham.ac.uk (Isaac Triguero)

demonstrate that designing spatial attention methods for few-shot learning is a nontrivial task and our method has proven effective for it. We evaluate our method using both shallow and deeper networks on three widely used few-shot learning benchmarks, miniImageNet, tieredImageNet and CUB, and demonstrate very competitive performance.

Keywords: Few-shot learning, Meta-learning, Spatial attention, Saliency object detection, Adaptive pooling, Feature aggregation

1. Introduction

Deep learning techniques have achieved an impressive performance on artificial intelligence [30, 4]. However, humans have the ability to learn knowledge from very few examples, while deep learning approaches fail to do so, often requiring large numbers of examples to extract useful patterns. To fill the gap, the relatively new research field of few-shot learning has emerged targeted at learning quickly from a limited number of labelled examples [18, 39]. It has been widely studied in recent years within the computer vision community [72, 68, 19, 71, 59].

To learn a new visual concept from very few labelled images, conventional machine learning algorithms need to train repeatedly on the few available labelled examples, which may easily result in overfitting. Since humans quickly adapt already known knowledge to a new visual concept, a number of meta-learning approaches have been inspired by this to tackle few-shot learning problems [19, 59, 71]. To mimic human learning, these methods extract meta-knowledge from a large scale learning task or a collection of few-shot learning tasks, and then transfer this meta-knowledge to unseen few-shot learning tasks comprising novel categories. Meta-knowledge can be represented by various algorithm components, such as a general feature extractor [72, 68, 43], a distance metric [71], dynamic convolutional kernels [96], promising initial model parameters [19, 20, 87], optimisation strategies [60, 49], model parameter predictor [59, 26], or an example generator [16, 80, 12].

Besides transferring meta-knowledge to new learning tasks, humans can extract and concentrate on the object of an image, so that they can learn quickly without background interference. This ability is extremely useful when we learn from very few samples, because the few images available may include a lot of uncertainty, such as background clutters or different scales of objects. This ability can be achieved in either image space or embedding space when we use Convolutional neural networks (CNNs) to extract features. In image space, the approaches in [93, 83] used an object detection method to separate the foreground and background of an image for few-shot learning. In embedding space, the approach in [86] applied an attention mechanism to focus on the salient regions of an embedding. An embedding is an transformed representation of an image, which preserves the spatial relationships of features. However, none of the above methods leverage spatial attention in both the original image and embedding space. When the data is limited, we do not want to miss any useful spatial information. Therefore, performing spatial attention in both spaces potentially provides more information on where to focus in a visual learning process.

To our knowledge no existing few-shot learning approach addresses the problem with respect to the downsampling used within a network. Most techniques use off-the-shelf pooling techniques, such as max-pooling and average-pooling. The pooling operation is a process that loses information. This is not an issue for standard learning tasks, since we have plenty of training data and the testing classes are seen in the training set, the max or average pooling can work well with a well-trained robust feature extractor. However, this may hinder performance in few-shot learning. When labelled data is limited, training samples may not be representative. The trained convolutional block may extract irrelevant features, and therefore the subsequent pooling operation may lose relevant features (max-pooling) or mix up relevant and irrelevant features (average-pooling), which would affect classification performance. In addition, the commonly used pooling techniques perform pooling independently in different channels and ignore the correlation between the features at the same spatial

location in different channels, which may lose spatial importance. Therefore, a more appropriate pooling operation is needed to fuse local information for few-shot learning problems. The fusion process should take into account spatial importance and be applicable to novel classes.

To address these issues, we propose a fusion few-shot learning approach that incorporates spatial attention in both the original image space and embedding space. Specifically, in image space, we design a saliency object detection (SOD) module, in which an Edge Guidance Network (EGNet) [97] pre-trained on DUTS dataset [75] is employed to extract the saliency maps of samples, which are as an additional channel to the original RGB images. This can be seen as a pre-learning process that learns to tell the feature extractor where to focus. In embedding space, we design a spatial attention based adaptive pooling (Ada-P) module to replace the conventional pooling methods for few-shot learning, in which a learnable pooling weight generation block is trained to assign different pooling weights to the features at different spatial locations for each individual embedding. The module performs weighted pooling by taking into account the importance of the features at different spatial locations, which can pay more attention to the salient regions and avoid discarding useful information [93]. Since the sizes of receptive fields in different convolutional layers vary, we learn a specific pooling module for each convolutional layer. To consider the correlations between channels, we use CNNs as a meta-learner to assign pooling weights. Different from regular CNNs, our CNNs-based meta-learner is lightweight (only including one output channel) and can be incorporated into different CNNs-based few-shot learning approaches. Overall, our contribution is as follows.

- We are the first in the few-shot learning field to perform fusion spatial attentions in both original image space and embedding space, providing attentional information to the feature extractor on where to concentrate.
- We propose an adaptive pooling method for few-shot learning problems, which uses a meta-learner to learn a proper pooling operator that reduces

the loss of useful information when available data is limited. This module is lightweight and both Ada-P and SOD can be used as plug-and-play modules for varied few-shot learning problems.

- We compare the performance of the proposed approach against the state-of-the-art, testing its performance on 5-way 1-shot and 5-shot tasks on three popular few-shot learning benchmarks, including miniImageNet [60], tieredImageNet [61] and CUB [73] and achieve very competitive performance.

- We investigate the use of advanced spatial attention methods into few-shot learning to illustrate the novelty of our work and compare our methods with them. The results demonstrated designing spatial attention methods for few-shot learning is a nontrivial task.

The rest of this paper is structured as follows. Section 2 presents the related work on few-shot learning, attention mechanism, pooling methods and SOD. Section 3 describes a standard few-shot learning problem and introduces our proposed few-shot learning approach. Section 4 defines the experimental settings and Section 5 analyses the experimental results. Finally, conclusions and future works are discussed in Section 6.

2. Related work

This section briefly reviews related research fields to define and describe our proposal. First, we present the background on few-shot learning (Section 2.1). Then, we briefly cover three closely related fields: attention mechanisms (Section 2.2), pooling (Section 2.3) and SOD (Section 2.4).

2.1. Few-shot learning

Few-shot learning [18], targeting learning with few labelled examples, has been extensively studied in recent years [79]. Zero-shot learning [44, 46, 45] is a related research field aiming at learning without any example, but some

auxiliary information is provided, such as textual descriptions. Our focus in this paper is few-shot learning.

Generally speaking, existing approaches on few-shot learning can be classified into three main categories.

1. **Fast parameterisation based approaches** learn a general fast parameterisation strategy that can quickly fine-tune a learner or predict the parameters of a learner for each particular few-shot learning task. The work in [15] provided a strong fine-tuning baseline. Another representative method learned a promising parameter initialisation that can be quickly fine-tuned to different task-specific parameters [19]. A few extensions of this work have been proposed by taking into account the computational burden [2], optimisation strategies [60, 49], uncertainties [20, 89], meta-level overfitting problems [32], the heterogeneity and homogeneity of learning tasks [87, 64, 70], an update rule that preconditions gradients [21], the robustness regularisation [76], the representation change rather than representation reuse [55], and semi-supervised learning [48]. Some other approaches learned a meta-learner to predict the task-specific model parameters without fine-tuning. Most of them learned a general feature extractor and a meta-learner to predict the class-specific or task-specific parameters of the fully connected layer [59, 25, 58, 26, 70].
2. **Data generation based approaches** tackle few-shot learning by generating more synthetic data. Some approaches applied generative models, such as generative adversarial networks [27], to generate more artificial data to assist training [16, 80, 22, 95] or generate more auxiliary embeddings [9, 47]. Other methods augmented the few data by using different data augmentation algorithms, such as mixing foregrounds and backgrounds [93], image deformation [12], and jigsaw augmentation [11]. Note that the work in [93] also used a pre-trained saliency network to extract the foreground and background of an image. However, we use a different strategy to incorporate saliency maps into few-shot learning.

3. **Metric learning approaches** address few-shot learning by comparing the similarities in a learned metric space. Specifically, they learn a general feature extractor to transform the training and test examples into embeddings, then assigned a test embedding to its nearest training class based on Euclidean, cosine distance or a learned distance metric [68, 72, 71]. A number of research works have been built upon this baseline and progressed on theoretical analysis [7] or proposed various strategies for similarity comparison, such as task-specific scaled metric [56], separating foregrounds and backgrounds [83], finding task relevant features [43], introducing attention mechanism [86], using multiple prototypes to represent each class [3], comparing similarity in subspace [67], task-specific margin loss [42], and aggregating embeddings [88, 69]. Some other approaches constructed graphs based on the extracted embeddings and transformed the metric learning problems into label propagation or edge labelling problems using graph neural networks [52, 36, 23]

Although the existing approaches can achieve impressive results, as far as we know none have performed spatial attention in both original image and embedding space. Besides, they all used off-the-shelf max-pooling technique to downsample embeddings, which could cause more information loss in few-shot learning. Instead, we apply a SOD technique and a more tailored pooling strategy to conduct spatial attention in both image and embedding space, so that the feature extractor would pay more attention to the salient regions and avoid losing useful information.

2.2. Spatial Attention mechanisms

Spatial attention mechanisms [100] aim to let a machine learner focus on the relevant parts of the features, and have been widely applied in visual recognition [94, 98] and natural language processing systems [8]. Our Ada-P module uses a meta-learner to assign a specific pooling weight to the feature vector at each spatial location, which can be seen as a spatial attention mechanism. Our method is related to a few research works [84, 8, 74, 33, 53]. These methods applied

different strategies, such as using CNNs or multilayer perceptron [84, 8, 74, 53], performing downsampling, upsampling operations and residual connections [74], measure the compatibility between the final global feature and the features in the intermediate layers [33], on embeddings to assign weights to the features at different spatial locations. They performed attention after each convolutional block while our adaptive pooling module is incorporated in each convolutional block serving as a pooling operator. They mostly introduced much more trainable parameters while our Ada-P module is lightweight containing only one convolutional kernel. Their aim is to refine embeddings whereas our target is to find an appropriate way to downsample embeddings while preserving useful information. In addition, these works all tackled standard learning tasks with sufficient training data, while our focus is on few-shot learning problems.

2.3. Pooling

Pooling [6] is widely used in deep learning, summarising locally extracted features into statistics [57]. It is an important component in CNNs in order to reduce the number of parameters and computational burden, and improve the translation invariance of the network. The most commonly used pooling techniques are max-pooling and average-pooling, which downsample each sub-region by taking either the max or mean value of that sub-region. These two methods are simple and effective but have their own drawbacks. Max-pooling may lose useful information while average-pooling ignores the importance of relevant and irrelevant features. To address the issues, a few works have been proposed that explore a better way of pooling by theoretical analysis [6], using overcomplete rectangular pooling blocks [34], learning a linear combination of max and average pooling [40], considering overlapping between adjacent pooling regions [38], introducing detail-preserving image downscaling method [66], etc. The above approaches tackled standard recognition tasks, while in few-shot learning the shortcomings of max or average pooling are more noticeable. When the labelled data is limited, training samples may not be representative and the feature extractor may extract irrelevant features. The subsequent pooling operation may

lose relevant features (max-pooling) or mix up relevant and irrelevant features (average-pooling). Besides, the standard learning problems are evaluated on the same label space with the training set, which does not require too much adaptability, since the relevant features have already been seen during training. In contrast, few-shot learning tasks are tested on a different label space from the training phase, in this scenario, max or average pooling cannot provide too much adaptability. Therefore, a more general and proper pooling module for few-shot learning needs to be explored. Different from the above pooling methods, we take inspirations from meta-learning approaches for few-shot learning and propose a meta-learner that assigns pooling weights to features at different spatial locations for each individual embedding.

2.4. Salient object detection

Salient object detection aims to predict the most distinctive objects in an image, which has been widely applied to many object-level applications in various areas such as object recognition [65], object detection [92, 62], image retrieval [28], weekly supervised semantic segmentation [81, 78], image cropping [77] and image captioning [17, 14]. Normally, salient object detection includes two steps: 1) detecting the most salient object for a given image, 2) segmenting the salient object in this image and generating a binary saliency map indicating the locations of salient pixels. Early SOD models mostly rely on low level features [99, 35] or heuristic priors such as contrast [13] and background prior [82]. However, early SOD models are not robust enough to handle complicated scenarios since it is difficult for them to capture the high-level semantic information given the hand-craft features. Recently research into CNNs has significantly stimulated the development of SOD areas and many deep learning based SOD models have emerged. These SOD models are able to generate saliency maps accurately without any prior knowledge such as information on the background. In this paper, EGNet [97] was chosen to generate saliency maps as a recent approach offering competitive performance on standard SOD benchmarks.

3. Methodology

In this section we describe the proposed fusion spatial attention approach. We formulate the few-shot learning problem in Section 3.1. The overall pipeline of our few-shot learning approach is depicted in Section 3.2. Section 3.3 describes our SOD module. Section 3.4 presents our Ada-P module. Finally, Section 3.5 provides an overview of our training strategy.

3.1. Problem set-up

A typical N-way K-shot classification task classifies a test example into one of N unique classes based on K labelled training examples for each of N class. For each N-way K-shot classification task, the training set $D_{_{train}} = \{T_i\}_{i=1}^{N}$ includes N classes of the training data, in which $T_i = \left\{x_j^i, y^i\right\}_{j=1}^{K}$ contains K training examples. The test set $D_{_{test}}$ consists of N_q test examples whose labels belong to $D_{_{train}}$. A machine learner can be trained based on $D_{_{train}}$ to classify test examples, however, the limited training examples often lead to overfitting.

In a typical meta-learning setting, there are three meta-datasets, meta-training set $\mathfrak{D}_{meta-train}$, meta-validation set $\mathfrak{D}_{meta-validation}$ and meta-testing set $\mathfrak{D}_{meta-test}$. Each of them consists of a number of few-shot learning tasks, such as $\mathfrak{D}_{meta-train} = \left\{\left(D_{train}^j, D_{test}^j\right)\right\}_{j=1}^{N_{train}}$. There is no overlapping among their label spaces. Meta-learning approaches target to learn some transferable meta-knowledge from a meta-training set $\mathfrak{D}_{meta-train}$ and apply it to tackle unseen few-shot learning tasks in the meta-testing set $\mathfrak{D}_{meta-test}$. The meta-validation set is usually used to select hyper-parameters and the best performing model.

3.2. Few-shot learning pipeline

The workflow of our few-shot learning approach, as shown in Figure 1, is built upon ProtoNet [68]. Note that our backbone is not limited to ProtoNet and our SOD and Ada-P module can be incorporated into any few-shot learning pipeline. We choose ProtoNet as our backbone for several reasons. First, compared to fast parameterisation and data generation based approaches, ProtoNet

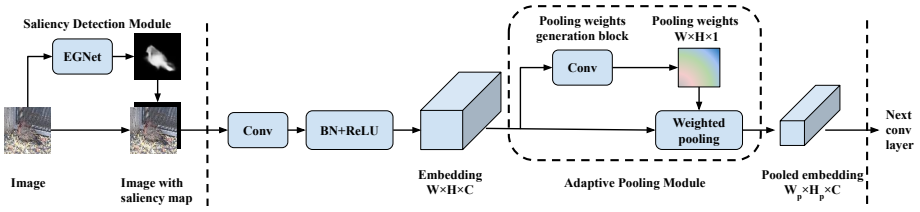

Figure 1: Workflow of the proposed SOD and Ada-P module. Conv represents convolutions. BN stands for batch normalisation [31]. In image space, a pre-trained saliency network is applied to extract the saliency map of an image, then it is attached to the image as an additional channel. In the embedding space, a pooling weights generation block is learned to assign a specific pooling weight map ($R^{W\times H\times 1}$) for each embedding ($R^{W\times H\times C}$); then weighted pooling is conducted based on the generated pooling weights per channel.

is computationally efficient. It does not require any inner optimisation or data generation process. Second, ProtoNet is simple and effective and has been used as a backbone for a few advanced few-shot learning approaches [80, 71, 69]. This makes it a fair to use as a backbone when comparing with other state-of-the-art methods. Third, from our experiments, ProtoNet can always achieve a relatively good and stable performance on different datasets. Our pipeline includes the following components: a SOD module, a feature extractor, a distance module, a loss function and an inference mechanism. The whole training procedure of our few-shot learning pipeline is presented in Algorithm 1.

1. **Saliency object detection module:** This module employs the pre-trained EGNet [97] to extract the saliency map s_j^i of an image x_j^i. A saliency map highlights the most visually distinctive objects in an image [97], which can guide the subsequent feature extraction. The extracted saliency map serves as an additional channel in the original image space alongside the RGB channels, which will be fed into the feature extractor as a whole $[x_j^i, s_j^i]$, $[,]$ representing concatenation along the last dimension.
2. **Feature extractor:** The feature extractor $f_{\{\phi,\psi\}}$ aims to transform an example into a high-level representation, ϕ and ψ are learnable parameters of CNNs and our Ada-P module, respectively. Given a training set

$D_{train} = \{T_i\}_{i=1}^{N}$, $T_i = \left\{x_j^i, y^i\right\}_{j=1}^{K}$, a testing set $D_{test} = \{(q_j, \widetilde{y}_j)\}_{j=1}^{N_q}$ and their corresponding extracted saliency maps $\left\{s_j^i\right\}_{j=1}^{K}$, $\left\{s_j^q\right\}_{j=1}^{N_q}$, the feature extractor $f_{\{\phi,\psi\}}$ transforms a training example $[x_j^i, s_j^i]$ and a test example $[q_j, s_j^q]$ into an embedding $f_{\{\phi,\psi\}}([x_j^i, s_j^i])$ and $f_{\{\phi,\psi\}}([q_j, s_j^q])$. f_ϕ is usually represented by CNNs or ResNet [29], in which several operators are stacked, such as convolutions, the batch normalisation (BN) [31], an activation function and a pooling operator. Following the idea of ProtoNet, the prototype of each class is the mean embedding of the training examples belonging to its class:

$$p_i = \frac{1}{K}\sum_{j=1}^{K} f_{\{\phi,\psi\}}([x_j^i, s_j^i]) \tag{1}$$

3. **Distance module:** This module applies a distance metric $\rho(\cdot)$ to measure the similarity between the embedding of a test example $f_{\{\phi,\psi\}}([q_j, s_j^q])$ and the prototype of each class p_i as $\rho(p_i, f_{\{\phi,\psi\}}([q_j, s_j^q]))$. The commonly used distance metrics are negative Euclidean distance, cosine distance, scaled Euclidean or cosine distance or a learned distance metric. In our approach, we use scaled Euclidean and cosine distance for shallow and deeper CNNs-based feature extractors, respectively. These settings are widely used in preivours works [56, 25].
4. **Loss function:** We choose cross-entropy loss to train each task. First, the softmax function is applied over the distance between the test embeddings and the prototypes as follows:

$$p_{\{\phi,\psi\}}(y = y^i \mid [q_j, s_j^q]) = \frac{\exp(\rho(p_i, f_{\{\phi,\psi\}}([q_j, s_j^q])))}{\sum_{i'} \exp(\rho(p_{i'}, f_{\{\phi,\psi\}}([q_j, s_j^q])))} \tag{2}$$

Then, the loss function can be formulated as

$$L(\phi, \psi) = -\frac{1}{N_q}\sum_{j=1}^{N_q} \log\, p_{\{\phi,\psi\}}(y = \widetilde{y}_j \mid [q_j, s_j^q]) \tag{3}$$

where N_q is the number of test examples in each training task, $\widetilde{y}_j$ is the true label of q_j.

5. **Inference:** The inference is conducted on the meta-testing set $\mathfrak{D}_{meta-test}$, whose label space has no overlapping with the meta-training set $\mathfrak{D}_{meta-train}$. The procedure is nearly the same with the meta-training phase through the trained feature extractor, distance module and a softmax function. Then, a test example can be classified into one class by taking its highest probability:

$$l_t = \underset{i}{\operatorname{argmax}}\, p_{\{\phi,\psi\}}(y = y^i \mid x_{qt}) \tag{4}$$

where x_{qt} is a test example in a meta-testing task and l_t is its predicted label.

3.3. Saliency object detection moduel

The SOD module aims to extract the salient object from an image and use it to guide the subsequent feature extraction. We employ a promising saliency object detection method, EGNet, which explicitly models complementary salient object information and salient edge information within a single network to preserve object boundaries and localise salient objects simultaneously. To leverage the extracted saliency maps for few-shot learning, we have explored several avenues to incorporate saliency maps into few-shot leaning approaches, which are presented in Section 5.1. Based on our experiments, we found providing the extracted saliency map as an additional channel alongside RGB channels offers the best performance. Therefore, we choose this method to be our main avenue of leveraging saliency maps for few-shot learning. Note that the work in [93] also used a pre-trained saliency network to extract the foreground and background of an image. They designed a complex module, which includes much more convolutional layers than our method, to mix the foregrounds and backgrounds from a batch of images. While we treat a saliency map as an additional image channel and further employs Ada-P modules to perform spatial attention in embedding space. Compared to their approach, our Ada-P module is more lightweight and our avenue of utilising saliency maps is simpler, while we achieve similar or even better performance on some few-shot learning tasks.

3.4. Adaptive pooling module

The Ada-P module can be placed in the feature extractor after each convolutional operation as shown in Figure 1. The pooling weight generation block learns to adaptively generate feature fusing weights for each embedding. Then local features in a sliding window can be fused based on the generated weights by taking into account the spatial importance. Let $E \in R^{W \times H \times C}$ be an embedding serving as the input of a convolutional layer, where W, H, C represent the width, height of feature maps and the number of channels, respectively. After a standard convolutional operation followed by BN and an activation function, we obtain $\widetilde{E} = Conv(E)$, $\widetilde{E} \in R^{W \times H \times C}$. Then, the embedding $\widetilde{E}$ is fed into the pooling weight generation block $g_\psi(\cdot) : R^{W \times H \times C} \rightarrow R^{W \times H \times 1}$ to generate the adaptive pooling weights for the features at different spatial locations:

$$w = g_\psi(\widetilde{E}) \div t^2, w \in R^{W \times H \times 1} \tag{5}$$

Specifically, our pooling weight generation block is represented by a convolutional layer with a single convolutional kernel, which is much more lightweight compared to the convolutional operation in the feature extractor. We choose convolutions to be the meta-learner because it provides a larger receptive field and considers the features in all channels at the same spatial location. Furthermore, g_ψ also includes a BN layer followed by a sigmoid function. We use a sigmoid function to limit the pooling weight values between 0 and 1. We further divide the generated weights by t^2 for quick convergence, t is the pooling window size.

Then, weighted pooling is conducted over the embedding $\widetilde{E}$ to fuse local features. The generated pooling weights w are shared among different channels. Therefore, for each pooling sub-region Ω_k whose size is determined by the pooling window size, the weighted pooling can be computed as:

$$\widetilde{E}_{pooled}[k] = \sum_{i \in \Omega_k} w[i] \widetilde{E}[i] \tag{6}$$

where $\widetilde{E}_{pooled}[k] \in R^{1 \times 1 \times C}$ is the pooled embedding for sub-region Ω_k, $w[i] \in R^{1 \times 1 \times 1}$ and $\widetilde{E}[i] \in R^{1 \times 1 \times C}$ are the generated pooling weight and the embedding

at $i-th$ location in sub-region Ω_k. After performing weighted pooling for all the sub-regions, the fused embedding $\widetilde{E}_{pooled}$ will feed into the next convolutional layer. Pseudocode for our Ada-P module is provided in Algorithm 2.

It is noteworthy that our adaptive pooling module is not limited to be used in metric learning approaches described in Section 3.2. It can also be applied in other methods, such as fast parameterisation approaches as a plug-and-play module, as we will show in Section 5.

3.5. Training strategies

The training strategies of the existing meta-learning approaches can be broadly categorised into two classes, namely episode-based training [68, 19] and large scale training [59, 70]. Both train on the meta-training set, which is a fair test. We adopted both of them for training the models with different architectures.

1. **Episode-based training** aims to imitate the learning processes in meta-testing during meta-training. In each meta-training iteration, a number of N-way K-shot learning tasks $\left\{\left(D^j_{train}, D^j_{test}\right)\right\}_{j=1}^{N_m}$ are sampled from the meta-training set $\mathfrak{D}_{meta-train}$, N_m is the meta-batch size. The meta-training is performed based on the tasks sampled on the fly, which can be seen as episodes. We adopt episode-based training strategy for 4-convolutional-layers models for fair comparison with other methods.
2. **Large scale training** utilises the meta-training set as a whole. Specifically, it adds a fully-connected (FC) layer on the feature extractor and classifies all the classes in the meta-training set simultaneously based on all the available training examples of each class, which is similar to the pre-training on ImageNet for recognition tasks. After the large scale training, the FC layer is removed, the feature extractor can be directly applied to few-shot classification tasks or be fine-tuned for a few epochs by episode-based training. This large scale training usually provides better performance on a deep model. Therefore, we adopt the large scale training strategy for ResNet models.

Algorithm 1 The training procedure of our few-shot learning pipeline

Input: Meta-training set $\mathfrak{D}_{meta-train}$

Input: The number of classes N, the number of training exampels K and the number of test samples N_q in each task, the number of convolutional layers C, the ProtoNet-based classifier P based on equation (1) and (2), the pre-trained SOD model SOD($\cdot$).

randomly initialize φ and ϕ
while not done **do**
  Randomly sample a batch of tasks T_i from $\mathfrak{D}_{meta-train}$
  $\mathcal{L} = 0$ //Initialise loss
  for all T_i **do**
    Randomly sample D^i_{train}, D^i_{test} for T_i, $D^i_{train} = \left\{\left(x^i_j, y^i_j\right)\right\}_{j=1}^{N\times K}$, $D^i_{test} = \left\{\left(x^i_i, y^i_i\right)\right\}_{j=1}^{N_q}$
    for all x^i_j in D^i_{train} and D^i_{test} **do**
      $s^i_j = \text{SOD}(x^i_j)$ //Obtain the saliency map of each sample
      $E^i_j = [x^i_j, s^i_j]$ //Add a saliency map as an additional channel
      for $c = 1$ **to** C **do**
        $\widetilde{E}^i_j = Conv_{\phi_c}(E^i_j)$ //Feed forward each Conv block
        $\widetilde{E}^i_{j,pooled} = \text{Ada-P}_{\psi_c}(\widetilde{E}^i_j)$ //Feed forward Ada-P module
        $E^i_j = \widetilde{E}^i_{j,pooled}$
      end for
    end for
    $E^i_{train} = \left\{E^i_j\right\}$, E^i_j is derived from $x^i_j \in D^i_{train}$ //Training embeddings
    $E^i_{test} = \left\{E^i_j\right\}$, E^i_j is derived from $x^i_j \in D^i_{test}$ //Testing embeddings
    $\mathcal{L}^i_{test} = \sum_{j=1}^{N_q} \mathcal{L}(P(E^i_{train}, E^i_j), y^i_j)$, $E^i_j \in E^i_{test}$, $y^i_j \in D^i_{test}$ //Compute loss for each task
    $\mathcal{L} = \mathcal{L} + \mathcal{L}^i_{test}$ //Accumulate losses for a batch of tasks
  end for
  Update ϕ and ψ based on $\bigtriangledown_{\phi,\psi}\mathcal{L}$ //Update the parameters of feature extractor and Ada-P module
end while

Algorithm 2 Ada-P module

Input: Embedding $\widetilde{E} \in R^{W \times H \times C}$, pooling region $\Omega = W \times H$, pooling weights generation block g_ψ, window size t, strides s, padding p

Output: $\widetilde{E}_{pooled}$

1: $w = g_\psi(\widetilde{E}) \div t^2, w \in R^{W \times H \times 1}$ //Obtain adaptive pooling weights

2: **for** Ω_k in Ω **do**

3: $\widetilde{E}_{pooled}[k] = \sum_{i \in \Omega_k} w[i]\widetilde{E}[i]$, $\widetilde{E}_{pooled}[k] \in R^{1 \times 1 \times C}$ //Weighted pooling for each sub-region

4: **end for**

4. Experimental framework

In this section, we present the experimental framework. Details of the three widely used datasets we use for few-shot learning as well as a dataset for SOD are described in Section 4.1. We provide an enumeration of the selected comparison algorithms and the commonly used network architecture of feature extractors in Section 4.2. We describe the meta-learning based evaluation approach in Section 4.3, including the few-shot learning tasks chosen for evaluation, evaluation and statistical measurements, and training settings.

4.1. Datasets

We evaluate our methods on three widely studied few-shot learning datasets, miniImageNet, tieredImageNet and CUB. Note that Omniglot [39] is also a well-known benchmark for few-shot learning. However, we do not choose it because recent methods have achieved nearly 100% accuracy, leaving limited space for improvement. The EGNet in SOD module is pre-trained on DUTS dataset for SOD tasks as per [97].

1. **miniImageNet** was proposed by [72] derived from the original ILSVRC-12 dataset [63]. It comprises 100 classes of colour images with 600 of each (60,000 in total). In our experiments, we use the widely used splits proposed by [60], which divides the 100 classes into 64 for meta-training,

16 for meta-validation and 20 for meta-testing. All the input images are resized to 84×84 as done by most few-shot learning approaches [60, 68, 19].

2. **tieredImageNet** was proposed by [61], which is a larger subset of ILSVRC-12 dataset [63] consisting of 608 classes of colour images (779,165 in total). These classes are grouped into 34 broader categories based on the higher-level nodes in the ImageNet hierarchy, in which 20 of them are used for meta-training, 6 for meta-validation and 8 for meta-testing. Therefore, there are 351, 97 and 160 classes for meta-training, meta-validation and meta-testing in total. All the images are resized to 84×84 following the existing approaches [61]. This dataset is more challenging and realistic compared to miniImageNet, since the meta-training and meta-testing set are less similar in the semantic space.
3. **CUB**: CUB-200-2011 proposed by [73] is an image dataset with photos of 200 bird species. It is comprised by 200 classes of colour images (11,788 in total). In our experiments, we use 100 of them for meta-training, 50 for meta-validation and 50 for meta-testing, following the split proposed by [10]. All the images are resized to 84×84 as done by [10]. Since the dataset only contains bird species, the few-shot learning tasks on this dataset can be seen as fine-grained classification tasks.
4. **DUTS**: DUTS [75] is the largest salient object detection benchmark, which includes 10,553 training images and 5,019 testing images. Most images are challenging with various locations and scales. We pre-train EGNet based on this dataset. All the images are resized to 84×84 to keep them the same size as those in datasets for few-shot learning. Note that there are a few overlappings, 245 images, between the meta-testing set of miniImageNet and the training set of DUTS. To strictly follow the meta-learning setting that the meta-testing samples should be unseen, we exclude those overlapping images from DUTS when pre-training EGNet for miniImageNet.

4.2. Comparison algorithms and network architecture

The existing few-shot learning approaches are not strictly comparable, since they use networks with different depths as a feature extractor or different training strategies. Since the few-shot learning community focuses on the effectiveness of the meta-learning strategy rather than the complexity of networks, we choose two widely used network architectures, 4Conv and Res12, as a feature extractor and select representative state-of-the-art methods based on the same network architecture for comparison. We also include several recent methods using deeper networks, which can further demonstrate the superiority of our SOD and Ada-P module.

Concretely, for 4Conv based feature extractor, we choose MAML [19], A2P [59], MetaOptNet [41], R2D2 [5] from fast parameterisation based approaches, MetaGAN+RN [95], SalNet [93] from data generation based approaches, ProtoNet [68], RN [71], L2AE-D [69], TPN [52], GNN [23] from metric learning approaches. For Res12 based feature extractor, we compare with LEO [64], A2P [59], MetaOptNet [41], MTL [70] from fast parameterisation based approaches, SNAIL [54], TADAM [56], DC [50], CTM [43], ProtoNet [68] from metric learning approaches. Note that we only compare against methods that provide results on a given dataset, therefore the selected comparison algorithms on different datasets can be different. Besides, we would like to acknowledge that comparing our SOD module with those methods that do not use saliency maps may not be completely fair, as they are not using exactly the same data, network architectures or experimental settings. However, our goal with the SOD module is to demonstrate that it is beneficial and compatible with them.

To demonstrate that Ada-P is particularly suitable for few-shot learning, we compare our module with several widely used and advanced pooling methods, namely max-pooling [6], average-pooling [6], overlapping-pooling [38], stochastic-pooling [90], mixed-pooling [40] and gated-pooling [40].

The architecture of EGNet is described in [97]. We slightly modify its architecture by removing the max-pooling layer and setting the stride of second layer as 1 to make it compatible with lower resolution images (84×84) in few-shot

learning datasets. The detailed architecture of 4Conv and Res12 are described as follows:

1. **4Conv** consists of 4 convolutional blocks. Each block is composed of a 3×3 convolution with 64 filters, followed by BN, a ReLU nonlinearity and a 2×2 max-pooling. In our approach, we replace max-pooling by our Ada-P module, in which the pooling weight generation block contains a 3×3 convolution with 1 filter, followed by BN and a sigmoid function. The pooling window size is 2×2. Following [41], we also include a dropblock [24] layer after each convolutional block to reduce meta-level over-fitting. The keep rate and block size are set as 0.85 and 3 respectively for all 4 layers on miniImageNet and tieredImageNet. For the distance metric, we choose scaled Euclidean distance following [88], the scale is set 64.
2. **Res12** consists of 4 residual blocks, each of them is composed of 3 convolutional layers. Max-pooling is performed after each residual block while in our approach we use adaptive pooling module, in which the pooling weight generation block contains a 3×3 convolution with 1 filter, followed by BN and a sigmoid function. Each convolutional layer in the residual block consists of a 3×3 convolution with k filters, followed by BN, a Leaky ReLU (0.1) nonlinearity. k is set to be 64 in the first residual block and is doubled every next block. A dropblock layer is added after each residual block following [41] to reduce meta-level overfitting. The keep rate and block size are the same as those for 4Conv model. Moreover, we choose cosine distance as distance metric following [25], since we found that cosine distance works better with Res12 model empirically.

4.3. Meta-learning evaluation

We evaluate our method on 5-way 1-shot, 5-way 5-shot learning tasks on all three datasets. Like most approaches based on 4Conv [68, 19, 52], we train our 4Conv based approach in episodic manner for a fair comparison. For training Res12 based approaches, we adopted large scale training strategy following [70, 88, 64, 43] for better performance. During meta-testing, the previous works

mostly evaluated their methods on 600 or 1,000 randomly sampled C-way K-shot classification tasks from the meta-testing set. To get a more reliable result, we evaluate our approach on 6,000 randomly sampled tasks for all three datasets. The average classification accuracy and 95% confidence interval are reported. All experiments are performed using TensorFlow [1] on a Titan Xp GPU.

1. **Episode-based training:** During meta-training, the meta-batch size is set as 3 for both 1-shot and 5-shot tasks. Note that a larger meta-batch size could contribute to faster convergence while we set our meta-batch size taking into account the limitation of GPU memory. For each few-shot task, besides the $N \times K$ training examples, we randomly sample 6 test examples per class to compute the meta-training loss. Following [68, 19, 52], we train our model with Adam [37] with an initial learning rate of 0.001. On miniImageNet and tieredImageNet, we train 300,000 episodes for 1-shot tasks and 200,000 episodes for 5-shot tasks. The learning rate is cut in half every 60,000 and 40,000 episodes for 1-shot and 5-shot tasks, respectively. On CUB dataset, we train 100,000 episodes for both 1-shot and 5-shot tasks and cut the learning rate in half every 20,000 episodes.
2. **Large scale training:** Following [70, 88, 43], we perform large scale training on Res12 using the whole meta-training set. A FC layer is added to the Res12 based feature extractor. Then the meta-training process is transformed into a 64, 351 and 100 classes classification problem for miniImageNet, tieredImageNet and CUB, respectively. Following [70, 41], we train our model with stochastic gradient descent with Nesterov momentum of 0.9 with an initial learning rate of 0.1 for 30,000 episodes on miniImageNet and CUB, 100,000 episodes on tieredImageNet. The learning rate is divided by 10 every 10,000 episodes for miniImageNet and CUB, 20,000 episodes for tieredImageNet. The batch size is set as 128. The weight decay is set to be 0.0005. We adopt random horizontal flip and random crop data augmentations as in [41, 43, 70]. After the large scale training, the feature extractor can be used in few-shot classification tasks without

any fine-tuning.

5. Results and analysis

In this section, we analyse the results obtained from different experimental studies. Specifically, our aims are:

- To explore and compare different methods of leveraging the extracted saliency maps for few-shot learning (Section 5.1).
- To compare the performance of our Ada-P module with several widely used and advanced pooling methods in few-shot learning problems (Section 5.2).
- To perform ablation studies that tests whether pooling operations are really needed in few-shot learning, verifies the working of Ada-P is not caused by adding more parameters (Section 5.3).
- To analyse the benefits and flexibility of our SOD and Ada-P module when they are incorporated into existing few-shot learning approaches (Section 5.4).
- To check whether the superiority of our few-shot learning approach is maintained on various datasets (small, large, fine-grained datasets) based on both shallow and deep models (Section 5.5).
- To illustrate the novelty of designing effective spatial attention methods for few-shot learning problems, comparing the performance of our Ada-P and SOD modules against advanced spatial attention methods (Section 5.6).

5.1. Comparisons with different methods of leveraging saliency maps

To maximise the benefits of using saliency maps for few-shot learning, we explore a few avenues of incorporating them: (a) adding a saliency map as an additional channel alongside RGB channels; (b) normalising a saliency map between 0 and 1, then removing background by multiplying the saliency map

with each of RGB channels; (c) using 2 networks to extract features from a saliency map and an image respectively, then adding the saliency embedding to the image embedding in each layer; (d) using 2 networks to extract features from a saliency map and an image respectively, then multiplying the image embedding by the saliency embedding in each layer. We choose ProtoNet as a baseline, since it is a widely used effective method for few-shot learning. To compare each approach we trained a 4Conv network on the miniImageNet dataset and report final testing performance. We present a comparison of each method in Table 1. Removing the background by pre-multiplying with the saliency map produces a marked drop in performance, likely due to the loss of additional contextual information, and the fact that the feature extractor focuses more on the outline rather than the texture of a salient object. Performance across other techniques is closer, but adding a saliency map as an additional channel offers the best performance on both 1-shot and 5-shot tasks. Therefore, we choose this method to incorporate saliency maps in most of our experiments.

Table 1: Comparisons with several methods of leveraging saliency maps for few-shot learning. **Archt.** represents the architecture of the feature extractor.The last number of **Archt.** stands for the number of filters in each convolutional layer. The average accuracy (%) with 95% confidence intervals are reported. $^{+}$ represents an enhanced version of ProtoNet.

		miniImageNet 5-way	
Methods	**Archt.**	**1-shot(%)**	**5-shot(%)**
ProtoNet^{+} [68]	4Conv-64	52.34 ± 0.26	69.90 ± 0.18
ProtoNet^{+} [68] - (a) adding 1 channel	4Conv-64	55.00 ± 0.26	72.77 ± 0.18
ProtoNet^{+} [68] - (b) removing background	4Conv-64	46.28 ± 0.26	66.48 ± 0.18
ProtoNet^{+} [68] - (c) 2 networks - add	4Conv-64	54.39 ± 0.26	72.73 ± 0.18
ProtoNet^{+} [68] - (d) 2 networks - mul	4Conv-64	54.43 ± 0.26	71.33 ± 0.18

5.2. Comparisons with pooling baselines

To demonstrate the superiority of Ada-P, we first compare our Ada-P with a few pooling baselines on few-shot learning problems in Table 2. The same

as before, we choose ProtoNet as the baseline. We compare Ada-P with a few pooling baselines mentioned in Section 4.2. From Table 2, we can see that our approach outperforms all these baselines, which verifies Ada-P is more suitable for few-shot learning problems.

Table 2: Comparisons with several pooling baselines. **Archt.** represents the architecture of the feature extractor.The last number of **Archt.** stands for the number of filters in each convolutional layer. The average accuracy (%) with 95% confidence intervals are reported. $^+$ represents an enhanced version of ProtoNet.

		miniImageNet 5-way	
Methods	**Archt.**	**1-shot(%)**	**5-shot(%)**
ProtoNet$^+$ [68] w/ Max-pooling	4Conv-64	52.34 ± 0.26	69.90 ± 0.18
ProtoNet$^+$ [68] w/ Avg-pooling	4Conv-64	53.01 ± 0.26	70.12 ± 0.18
ProtoNet$^+$ [68] w/ Max-pooling-overlap [38]	4Conv-64	53.23 ± 0.26	70.14 ± 0.18
ProtoNet$^+$ [68] w/ Avg-pooling-overlap [38]	4Conv-64	52.59 ± 0.26	70.43 ± 0.18
ProtoNet$^+$ [68] w/ Stochastic-pooling [90]	4Conv-64	51.98 ± 0.26	69.92 ± 0.20
ProtoNet$^+$ [68] w/ Mixed-pooling [40]	4Conv-64	52.88 ± 0.26	70.87 ± 0.18
ProtoNet$^+$ [68] w/ Gated-pooling [40]	4Conv-64	53.16 ± 0.26	70.77 ± 0.18
ProtoNet$^+$ [68] w/ Ada-P	4Conv-64	**54.75 ± 0.26**	**71.63 ± 0.20**

To further show how our adaptive pooling works in comparison to commonly used pooling methods, we visualise the embeddings after the pooling operation in some convolutional layers of a few samples. Specifically, we use two ways to visual convolutional feature maps. First, we simply compress the multiple channels into a single channel by mean operation as shown in Figure 2. Second, we apply a famous CNNs based visualisation technique [91] by mapping the convolutional features to the input pixel space as shown in Figure 3. Figures 2 and 3 show the visualisation of the training embeddings in the first three convolutional layers using different pooling methods from a 5-way 1-shot task. The categories are trifle, Alaskan malamute, vase, lion and hourglass from left to right. We can see that in the first layer, the embeddings of max-pooling and avg-pooling look similar. These two pooling methods seize a few relevant features,

such as the shape of the Alaskan malamute and the vase, the eyes and noses of the lions, while our Ada-P module preserves more details, such as the hair features of the Alaskan malamute and the lions. For few-shot learning, we do not want to lose any useful information from the beginning when the training data is very small. Therefore, the end-to-end trained Ada-P module in the first layer learns to preserve more details, which is more suitable for few-shot learning. In the second and third layers, the feature maps are downsampled to a low-resolution space and the blurred mean feature maps in Figure 2 cannot provide much insight. However, taking advantage of the visualisation technique in [91], we can map the low-resolution feature maps back to the input pixel space. In Figure 3, we can see that the embeddings of Ada-P in the third layer focus more on the target objects, while the other pooling methods include more background noise. We can easily recognise the target object from the embeddings of Ada-P in the third layer compared to the original images. However, the embeddings of max-pooling and avg-pooling look blurred and it is difficult to distinguish the target object and the background. Therefore, we can conclude the Ada-P in the higher layers learns to perform spatial attention and suppresses background noise, which is aligned with our motivation.

5.3. Ablation studies on Ada-P

We further conduct two ablation studies shown in Table 3 to show the actual working of our Ada-P. Specifically, we first remove the pooling operations in ProtoNet and set convolutional stride as 2 to perform downsampling. It can be seen that all other pooling methods including ours outperform convolutional stride based downsampling, which shows that pooling is necessary for few-shot learning. Besides, we test whether our improvements are caused by adding more learnable parameters in CNNs. Since our meta-learner only includes one convolutional kernel, we add one more kernel in each convolutional layer in ProtoNet with max-pooling and average-pooling for a fair comparison. The results show our method outperforms ProtoNet with additional kernels significantly, which demonstrates that our improvements are not caused by adding more parameters

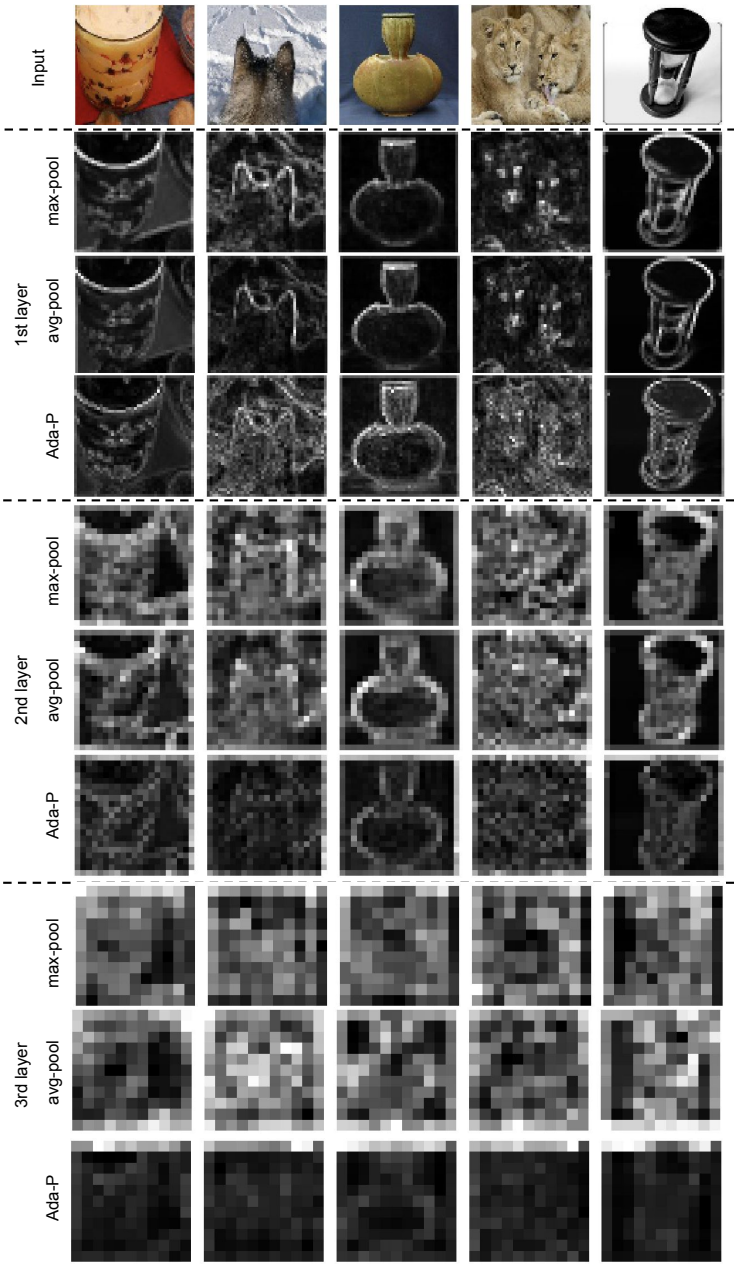

Figure 2: Visualisation of feature maps by compressing multiple feature maps into a mean one. The first row shows five input images. The Figures between dash lines represent the embeddings of different pooling operations in different layers.

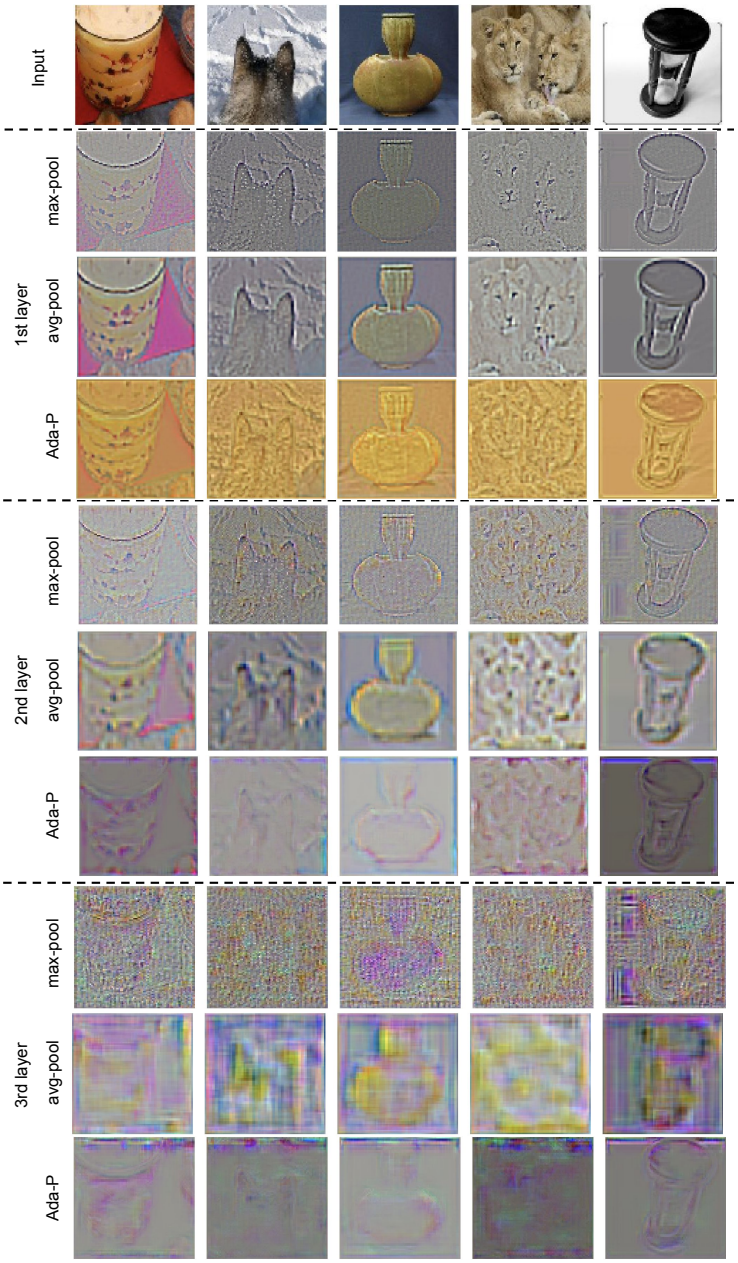

Figure 3: Visualisation of feature maps by Deconvolution [91]. The first row shows five input images. The Figures between dash lines represent the embeddings of different pooling operations in different layers.

but learning to assign adaptive pooling weights.

Table 3: Ablation study. **Archt.** represents the architecture of the feature extractor.The last number of **Archt.** stands for the number of filters in each convolutional layer. The average accuracy (%) with 95% confidence intervals are reported. $^+$ represents an enhanced version of ProtoNet.

		miniImageNet 5-way	
Methods	**Archt.**	**1-shot(%)**	**5-shot(%)**
ProtoNet$^+$ [68] w/ Conv-stride(2)	4Conv-64	49.98 ± 0.25	66.91 ± 0.17
ProtoNet$^+$ [68] w/ Max-pooling	4Conv-65	52.59 ± 0.26	70.37 ± 0.18
ProtoNet$^+$ [68] w/ Avg-pooling	4Conv-65	53.13 ± 0.26	70.67 ± 0.18
ProtoNet$^+$ [68] w/ Ada-P	4Conv-64	**54.75 ± 0.26**	**71.63 ± 0.20**

5.4. Incorporating Ada-P and SOD into existing approaches.

To verify the effectiveness and compatibility of our SOD and Ada-P module, we incorporate either and both of them into a few existing few-shot learning approaches based on a simple backbone, 4Conv. We choose MAML [19], RN [71], ProtoNet [68], L2AE-D [69] and TPN [52], because they are representative approaches from different few-shot learning branches and also include transductive and inductive methods. We perform comparison on miniImageNet using the 4Conv model. For ProtoNet, we use an enhanced version with dropblock and meta-batch training strategy. For the other methods, we do not add any training strategies described in Section 4.1, such as dropblock and augmentation. We reuse their released code and incorporate our SOD and Ada-p module, strictly following their respective experimental setting. Table 4 shows the comparison results. We can see our SOD and Ada-P module improves all the methods on 1-shot and 5-shot tasks, especially for MAML, it improves significantly by around 7%. When the Ada-P and SOD module is introduced individually, we can see all the approaches are improved. Note that SOD module improves more than Ada-P module on all the approaches. This is probably because the SOD module leverages auxiliary information (SOD datasets) to guide few-shot

learning, while Ada-P is only trained from a few-shot learning dataset. The auxiliary information can be seen as prior knowledge that provides more spatial information of an object, therefore, the SOD module generally performs better than Ada-P. When these two modules are incorporated together, we see that most of the tasks demonstrate further improvements compared to solely adding one of them. Since they can both be seen as a spatial attention mechanism, their positive influence on performance may overlap a bit. However, the SOD module is placed before the feature extractor to provide prior knowledge and Ada-P module is located in each convolutional layer to refine embeddings. Besides, the SOD module is pre-trained based on a SOD dataset and Ada-P is trained with the feature extractor based on few-shot learning dataset in an end-to-end manner. They should play a different role in few-shot learning, which is reflected by the visualisation in Figure 4 and 5, and we will provide more discussions in Section 5.6. Therefore, incorporating both of them can achieve better performance. Overall, this experiment verifies our Ada-P and SOD modules are beneficial for and compatible with different types of few-shot learning frameworks, including fast parameterisation and metric learning approaches or inductive and transductive methods.

5.5. Comparisons with state-of-the-art approaches on various datasets based on both shallow and deep models

In this section, we compare our method with state-of-the-art approaches on 5-way 1-shot and 5-shot classification tasks on various datasets using both 4Conv and Res12 models. The results and analysis are presented as follows.

Results on miniImageNet: The comparisons on miniImageNet using 4Conv and Res12 models are shown in Table 5. We choose comparable state-of-the-art methods from different branches for comparison. Based on the 4Conv feature extractor, our approach achieves state-of-the-art performance on 5-shot tasks and a comparable result to the best performance on 1-shot tasks. Note that SalNet [93] also utilised SOD in their approach, however, they incorporated a more complex network to mix foregrounds and backgrounds, introducing more

Table 4: Results after incorporating Ada-P and/or SOD into several existing approaches on miniImageNet. **Trans** represents if a method is a transductive method. BN represents a BN based transductive method. The last number of **Archt.** stands for the number of filters in each convolutional layer. The average accuracy (%) with 95% confidence intervals is reported. ↑ shows the improvements after incorporating Ada-P and/or SOD. $^+$ represents an enhanced version of ProtoNet.

			miniImageNet 5-way	
Methods	**Trans**	**Archt.**	**1-shot(%)**	**5-shot(%)**
MAML [19]	BN	4Conv-32	48.70 ± 1.84	63.74 ± 0.92
MAML [19] w/ Ada-P	BN	4Conv-32	50.74 ± 1.82 ↑ 2.04	66.85 ± 0.87 ↑ 3.11
MAML [19] w/ SOD	BN	4Conv-32	54.56 ± 1.82 ↑ 5.86	68.12 ± 0.87 ↑ 4.38
MAML [19] w/ SOD & Ada-P	BN	4Conv-32	55.76 ± 1.82 ↑ 7.06	70.11 ± 0.87 ↑ 6.37
RN [71]	BN	4Conv-64	50.44 ± 0.82	65.32 ± 0.70
RN [71] w/ Ada-P	BN	4Conv-64	50.95 ± 0.86 ↑ 0.51	66.46 ± 0.69 ↑ 1.14
RN [71] w/ SOD	BN	4Conv-64	54.02 ± 0.86 ↑ 3.58	67.81 ± 0.69 ↑ 2.49
RN [71] w/ SOD & Ada-P	BN	4Conv-64	53.95 ± 0.86 ↑ 3.51	68.59 ± 0.69 ↑ 3.27
ProtoNet$^+$ [68]	N	4Conv-64	52.34 ± 0.26	69.90 ± 0.18
ProtoNet$^+$ [68] w/ Ada-P	N	4Conv-64	54.75 ± 0.26 ↑ 2.41	71.63 ± 0.20 ↑ 1.73
ProtoNet$^+$ [68] w/ SOD	N	4Conv-64	55.00 ± 0.26 ↑ 2.66	72.77 ± 0.20 ↑ 2.87
ProtoNet$^+$ [68] w/ SOD & Ada-P	N	4Conv-64	57.29 ± 0.26 ↑ 4.95	74.60 ± 0.20 ↑ 4.70
L2AE-D [69]	BN	4Conv-64	53.85 ± 0.85	70.16 ± 0.65
L2AE-D [69] w/ Ada-P	BN	4Conv-64	55.12 ± 0.87 ↑ 1.27	70.55 ± 0.67 ↑ 0.39
L2AE-D [69] w/ SOD	BN	4Conv-64	56.16 ± 0.87 ↑ 2.31	72.00 ± 0.67 ↑ 1.84
L2AE-D [69] w/ SOD & Ada-P	BN	4Conv-64	56.90 ± 0.87 ↑ 3.05	73.06 ± 0.67 ↑ 2.90
TPN [52]	Y	4Conv-64	53.75 ± 0.86	69.43 ± 0.68
TPN [52] w/ Ada-P	Y	4Conv-64	55.19 ± 0.86 ↑ 1.44	70.90 ± 0.69 ↑ 1.47
TPN [52] w/ SOD	Y	4Conv-64	56.74 ± 0.86 ↑ 2.99	72.53 ± 0.64 ↑ 3.10
TPN [52] w/ SOD & Ada-P	Y	4Conv-64	57.08 ± 0.86 ↑ 3.33	73.68 ± 0.65 ↑ 4.25

convolutional layers, increasing the complexity when incorporating SOD beyond our approach. Instead, we perform a simpler method that adds a saliency map as an additional channel in image space and incorporates lightweight Ada-P modules, achieving similar performance on 1-shot tasks and a better result on

Table 5: Results on miniImageNet and tieredImageNet. The average accuracy (%) with 95% confidence intervals is reported. The best two performances are highlighted in bold. Res12 represents Res12 models and the behind number stands for the number of filters in the last residual block. WRN represents wide residual networks. Res18 represents ResNet-18 models. † uses the results of MetaOptNet with SVM trained only on the meta-training set. $^+$ represents an enhanced version of ProtoNet.

		miniImageNet 5-way		**tieredImageNet 5-way**	
Methods	**Archt.**	**1-shot(%)**	**5-shot(%)**	**1-shot(%)**	**5-shot(%)**
MAML [19]	4Conv-32	48.70±1.84	63.74±0.92	51.76±1.81	70.30±1.75
RN [71]	4Conv-64	50.44±0.82	65.32±0.70	54.48±0.93	71.32±0.78
MetaGAN+RN [95]	4Conv-64	52.71±0.64	68.63±0.67		
L2AE-D [69]	4Conv-64	53.85±0.85	70.16±0.65		
TPN [52]	4Conv-64	53.75±0.86	69.43±0.68	**59.91±0.94**	**73.30±0.75**
A2P [59]	4Conv-64	54.53±0.40	67.87±0.20		
SalNet [93]	4Conv-64	**57.45±0.88**	**72.01±0.67**		
ProtoNet$^+$ [68]	4Conv-64	52.34±0.26	69.90±0.18	52.44±0.27	70.91±0.23
[68] /w Ada-P & SOD	4Conv-64	**57.29±0.26**	**74.60±0.20**	**58.40±0.28**	**76.06±0.23**
SNAIL [54]	Res12-256	55.71±0.99	68.88±0.92		
TADAM [56]	Res12-512	58.50±0.30	76.70±0.30		
A2P [59]	WRN28	59.60±0.41	73.74±0.19		
LEO [64]	WRN28	61.76±0.08	77.59±0.12	66.33±0.05	81.44±0.09
MTL [70]	Res12-512	61.20±1.80	75.50±0.80		
DC [50]	Res12-512	62.53±0.19	79.77±0.19		
MetaOptNet† [41]	Res12-640	62.64±0.61	78.64±0.46	65.99±0.72	81.56±0.53
CTM [43]	Res18	**64.12±0.82**	**80.51±0.13**	**68.41±0.39**	**84.28±1.73**
ProtoNet$^+$ [68]	Res12-512	61.27±0.26	77.79±0.18	67.95±0.30	83.30±0.21
[68] w/ Ada-P & SOD	Res12-512	**63.29±0.27**	**80.10±0.19**	**70.28±0.30**	**84.92±0.21**

5-shot tasks. Based on the Res12 feature extractor, our approach achieves the second best performance on both 5-way 1-shot and 5-shot tasks. It is noteworthy that these methods are not strictly comparable since their network architectures are not exactly the same. For example, the best performing method, CTM, use ResNet-18 backbone, which represents a deeper architecture (11 million pa-

rameters) with around 37% more parameters than our Res12 model (8 million parameters). We apply the architecture that the majority of previous methods used and achieve competitive performance on both 1-shot and 5-shot tasks compared to the state-of-the-art results based on this backbone. Our method also improves upon ProtoNet$^+$ by around 2.0% on both 1-shot and 5-shot tasks. Since our approach is built upon ProtoNet$^+$ under the same experimental setting, these improvements also verify the effectiveness of our few-shot learning approach when using a deeper model.

Results on tieredImageNet: In Table 5, we also compare our approach with recent state-of-the-art methods that provide evaluations on 5-way 1-shot and 5-shot classification tasks on tieredImageNet using a 4Conv and Res12 based model. Note that the missing values in Table 5 indicate the methods are not tested on tiredImageNet. TieredImageNet is a more challenging benchmark compared to miniImageNet as discussed before, however, our approach still achieves a promising performance. Based on the 4Conv model, our approach achieves the best performance on 5-shot tasks and the second best on 1-shot tasks. The best performing method, TPN, is a transductive method that uses a few unlabelled examples to assist learning, while our method is an inductive approach that only uses labelled training examples to predict test examples. Based on the Res12 model, Ada-P achieves state-of-the-art performance on both 1-shot and 5-shot tasks, even compared to other approaches with a deeper model or more kernels. In addition, our method improves upon our baseline, ProtoNet$^+$, using both 4Conv and Res12 model.

Results on CUB: Finally, we test our approach on fine-grained few-shot classification tasks on CUB. Table 6 summarises the comparison of our and other few-shot learning approaches using both 4Conv and Res12 backbones. Note that some recent approaches achieve much higher performance on CUB based on a much deeper backbone, such as Res18 or WRN28. We do not include them in our comparison for fairness. From Table 6, we can see that our approach surpasses all the other methods by a large margin based on both shallow and deep backbones on both 1-shot and 5-shot tasks. They achieve

Table 6: Results on CUB. The average accuracy (%) with 95% confidence intervals is reported. The best two performances are highlighted in bold. $^+$ represents an enhanced version of ProtoNet. — indicates the method does not provide confidence interval. * represents results from [10].

		CUB 5-way	
Methods	**Archt.**	**1-shot(%)**	**5-shot(%)**
MatchingNet* [72]	4Conv-64	60.52 ± 0.88	75.29 ± 0.75
MAML* [19]	4Conv-64	54.73 ± 0.97	75.75 ± 0.76
RN* [71]	4Conv-64	62.34 ± 0.94	77.84 ± 0.68
ProtoNet$^+$ [68]	4Conv-64	57.61 ± 0.29	74.51 ± 0.18
ProtoNet$^+$ [68] w/ Ada-P	4Conv-64	62.40 ± 0.30	77.65 ± 0.18
ProtoNet$^+$ [68] w/ SOD	4Conv-64	**66.41 ± 0.30**	**81.86 ± 0.18**
ProtoNet$^+$ [68] w/ Ada-P & SOD	4Conv-64	**69.29 ± 0.30**	**83.86 ± 0.18**
MatchingNet* [72]	Res10-512	71.29 ± 0.87	83.47 ± 0.58
MAML* [19]	Res10-512	70.32 ± 0.99	80.93 ± 0.71
RN* [71]	Res10-512	70.47 ± 0.99	83.70 ± 0.55
ProtoNet$^+$ [68]	Res12-512	68.60 ± 0.26	85.51 ± 0.20
ProtoNet$^+$ [68] w/ Ada-P	Res12-512	70.22 ± 0.26	85.81 ± 0.20
ProtoNet$^+$ [68] w/ SOD	Res12-512	**72.47 ± 0.27**	**88.03 ± 0.20**
ProtoNet$^+$ [68] w/ Ada-P & SOD	Res12-512	**73.65 ± 0.27**	**88.39 ± 0.20**

further improvements when collaborating with each other, which demonstrates Ada-P and SOD are effective for fine-grained few-shot classification tasks.

5.6. Comparisons with other spatial attention methods on few-shot learning problems

Since both SOD and Ada-P modules can be seen as spatial attention mechanisms, to illustrate the novelty of our Ada-P and SOD modules for few-shot learning, we integrate a few recent representative spatial attention approaches into the few-shot learning framework and compare with our method. Our aim

is to show simply applying off-the-shelf spatial attention methods does not work satisfactorily for few-shot learning and we need to specifically design an effective approach. The selected advanced spatial attention methods are SCA [8], CBAM [84], Residual-AT [74], Interpret-SA [53], L2-pay-AT [33] from the computer vision field. It is noteworthy that these methods are not specifically designed for few-shot learning, some of them need to be tweaked a bit to be compatible with the few-shot learning framework. The same as before, we choose ProtoNet as a baseline and incorporate each spatial attention method into it respectively. To compare the meta-testing performance, we train a 4Conv backbone on miniImageNet dataset in an episodic manner. The results are shown in Table 7, where we can observe that simply incorporating the selected spatial attention methods into few-shot learning does not improve too much. SCA and Residual-AT even degrade the performance. They work well on standard learning tasks, while they may not be a good choice for few-shot learning tasks. This demonstrates designing an effective spatial attention method for few-shot learning is a non-trivial task. We can not simply introduce off-the-shelf spatial attention methods to improve few-shot learning. From Table 7, we can also see that our Ada-P and SOD modules outperform other spatial attention methods, respectively, on both 1-shot and 5-shot tasks, and incorporating both of them surpasses other methods by a large margin. This illustrates our Ada-P and SOD modules are especially effective for few-shot learning.

To better show how these spatial attention methods influence the feature extraction, we visualise the embedding space in the first two convolutional layers of a few samples as shown in Figure 4 and 5. The same as before, we compress the multiple channels into a single one by mean operation. From Figure 4, we can see that in the first layer the SOD module helps to outline the object from the background, while the Ada-P module preserves more details of the object. When incorporating both of them, the embeddings well preserve the shape and texture of the object and suppress the background noise. Other spatial attention methods either include more background noise or focus too much on a specific feature on the object, such as the eyes of lions. In the second layer, as

shown in Figure 5, our Ada-P module suppresses the background noise and the SOD module still helps emphasising the outline of the objects. Compared to other spatial attention methods, our Ada-P and SOD modules assist the feature extractor to focus more on the objects. We can clearly recognise the target object in the mean feature maps using our Ada-P and SOD modules while the embeddings of other methods look more blurred.

Based on the above analysis, we can conclude that simply incorporating other spatial attention methods into few-shot learning improves marginally, while our Ada-P and SOD modules are carefully designed and especially effective for few-shot learning problems, which demonstrates our contributions and novelty.

Table 7: Comparisons with several spatial attention methods on few-shot learning problems. **Archt.** represents the architecture of the feature extractor.The last number of **Archt.** stands for the number of filters in each convolutional layer. The average accuracy (%) with 95% confidence intervals are reported. $^+$ represents an enhanced version of ProtoNet.

		miniImageNet 5-way	
Methods	**Archt.**	**1-shot(%)**	**5-shot(%)**
ProtoNet$^+$ [68]	4Conv-64	52.34 ± 0.26	69.90 ± 0.18
ProtoNet$^+$ [68] w/ SCA [8]	4Conv-64	51.45 ± 0.26	68.83 ± 0.18
ProtoNet$^+$ [68] w/ CBAM [84]	4Conv-64	53.54 ± 0.26	70.19 ± 0.18
ProtoNet$^+$ [68] w/ Residual-AT [74]	4Conv-64	51.91 ± 0.26	70.35 ± 0.18
ProtoNet$^+$ [68] w/ Interpret-SA [53]	4Conv-64	53.70 ± 0.26	71.18 ± 0.18
ProtoNet$^+$ [68] w/ L2-pay-AT [33]	4Conv-64	52.52 ± 0.26	69.97 ± 0.18
ProtoNet$^+$ [68] w/ Ada-P	4Conv-64	54.75 ± 0.26	71.63 ± 0.20
ProtoNet$^+$ [68] w/ SOD	4Conv-64	55.00 ± 0.26	72.77 ± 0.20
ProtoNet$^+$ [68] w/ SOD & Ada-P	4Conv-64	57.29 ± 0.26	74.60 ± 0.20

5.7. Analysis summary and discussions

Based on the above results and analysis, we can conclude the following remarks, which also reflect the aims of different experiments at the beginning of

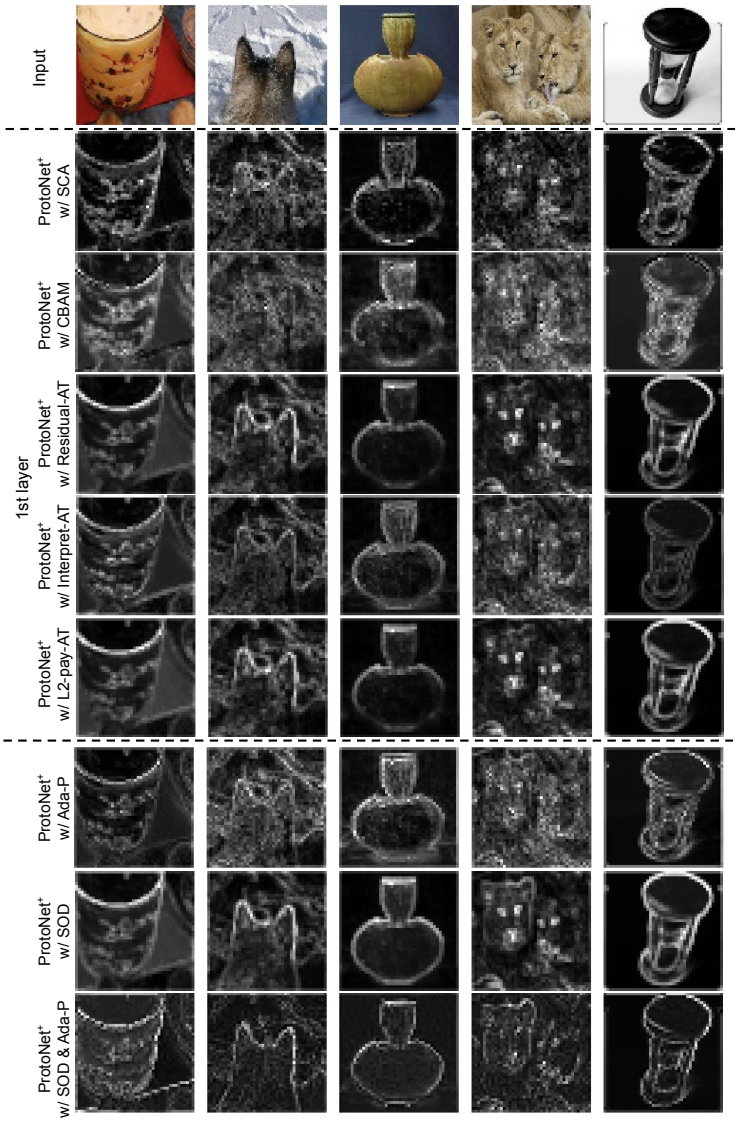

Figure 4: Visualisation of feature maps by compressing multiple feature maps into a mean one. The first row shows five input images. The Figures between dash lines represent the embeddings of different spatial attention methods in the first layer.

this section.

1. Adding a saliency map as an additional channel alongside RGB channels

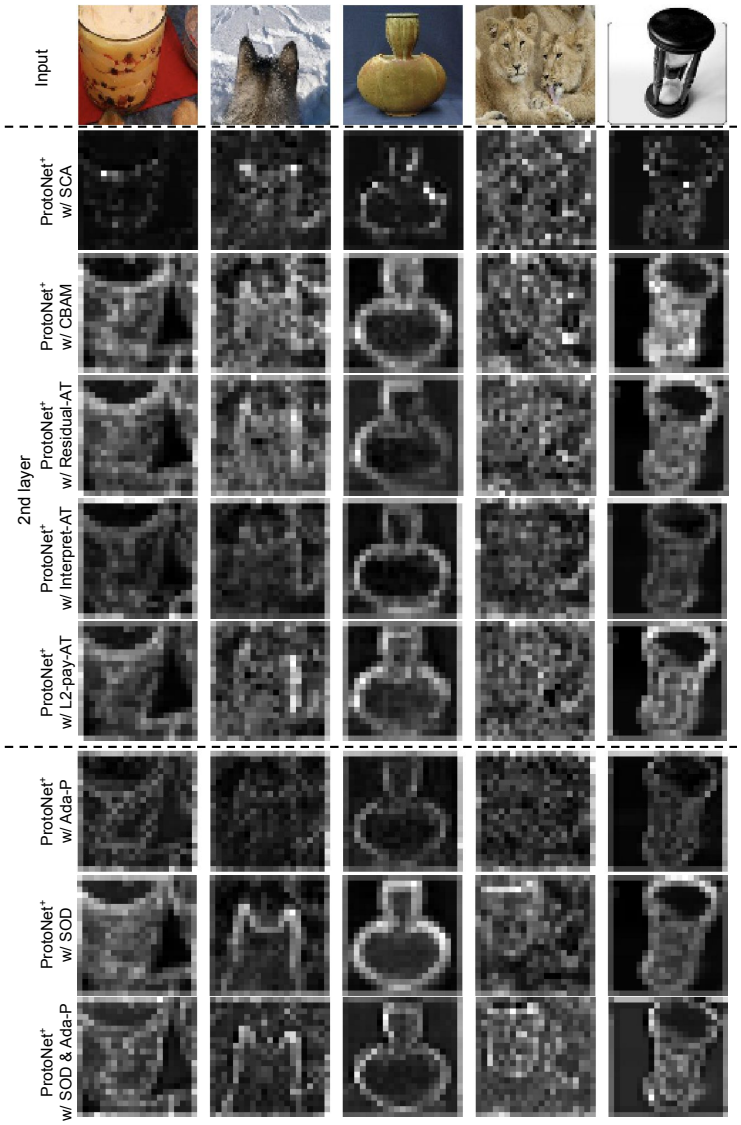

Figure 5: Visualisation of feature maps by compressing multiple feature maps into a mean one. The first row shows five input images. The Figures between dash lines represent the embeddings of different spatial attention methods in the second layer.

is the best way of leveraging saliency maps for few-shot learning, which is simple but effective.

2. Our Ada-P module performs better than a few pooling baselines in few-shot learning problems, such as max-pooling, average-pooling, overlapping-pooling, mixed-pooling and gated-pooling. This demonstrates our Ada-P is more suitable for few-shot learning.
3. The ablation studies show that pooling is a necessary component of a feature extractor for few-shot learning and the working of Ada-P is not simply caused by adding more learnable parameters.
4. Both of our SOD and Ada-P modules are compatible with and beneficial for most existing few-shot learning approaches. The results in Table 4 show significant improvements when incorporating Ada-P and SOD into representative few-shot learning methods.
5. The performance of our approach based on both shallow (4Conv) and deep (Res12) backbones on various datasets (small, large and fine-grained datasets) remains superior compared to the state-of-the-art approaches, which illustrates the effectiveness and robustness of our modules.
6. Both our Ada-P and SOD outperform advanced spatial attention methods on few-shot learning problems. The visualisation of the embedding space further shows our two modules help focus on target objects. These demonstrate our Ada-P and SOD modules are more effective for few-shot learning than other spatial attention methods.

Looking at the above summary, we point out several discussions, lessons learnt and future work directions as follows.

1. Currently, our Ada-P module concentrates on generating adaptive pooling weights, while a fixed-shape pooling window is applied as standard pooling methods. We argue that an adaptive-shape pooling window could provide more adaptability, which could further benefit few-shot learning problems. Therefore, we plan to explore the adaptive-shape pooling window along with the adaptive pooling weights for few-shot learning.
2. The experimental results show introducing saliency maps is helpful for few-shot learning. Some recent approaches incorporate some other additional

knowledge, such as semantic attributes [85, 42] or a category graph [51] and demonstrated their effectiveness. All these methods including ours aim to utilise additional knowledge to assist in learning a good representation, since few-shot learning problems suffer from a limited training data. We believe there would be some other additional knowledge beneficial for few-shot learning and also various ways to utilise it, which needs to be explored. Another potential future direction could be a fusion of all the available additional knowledge for few-shot learning.

3. In this work, we explore various avenues to incorporate the extracted saliency maps for few-shot learning. Rather than directly using them as an input, we could also utilise them as a supervisory signal to train a feature extractor to focus more on the salient region, so that during the meta-testing phase, there is no need to extract saliency maps for new images, which would potentially be more efficient.

4. In our SOD module, we choose EGNet as our saliency extraction method. As future work, we will explore and compare various saliency object detection approaches for few-shot learning on both high and low resolution images.

6. Conclusion

In this paper, we have presented a fusion spatial attention approach for few-shot learning problems, which introduces spatial attention in both original image space and embedding space. In image space, our SOD module extracts the saliency map of an image and provides it as an additional channel alongside the RGB image. In embedding space, our Ada-P module learns a meta-learner to assign adaptive pooling weights to the features at different spatial locations for each individual embedding. Our SOD and Ada-P modules can be used as a plug-and-play module and applied in various existing few-shot learning approaches, which has been demonstrated in experiments and achieved significant improvements. We have empirically demonstrated that using existing spatial

attention methods do not work well in few-shot learning, and our solution has shown to address that problem effectively. We evaluated our approach on three widely used benchmarks, miniImangeNet, tieredImageNet, CUB and achieved very competitive performance compared to state-of-the-art.

Acknowledgements

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

Compliance with ethical standards

Conflict of interest

The authors declare that they have no conflict of interest.

References

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, Michael, e.a., 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467 .

[2] Alex, N., Joshua, A., John, S., 2018. On first-order meta-learning algorithms. arXiv preprint arXiv:1803.02999 .

[3] Allen, K., Shelhamer, E., Shin, H., Tenenbaum, J., 2019. Infinite mixture prototypes for few-shot learning, in: International Conference on Machine Learning, pp. 232–241.

[4] Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al., 2020. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion 58, 82–115.

[5] Bertinetto, L., Henriques, J.F., Torr, P.H., Vedaldi, A., 2019. Meta-learning with differentiable closed-form solvers, in: International Conference on Learning Representations.

[6] Boureau, Y.L., Ponce, J., LeCun, Y., 2010. A theoretical analysis of feature pooling in visual recognition, in: International Conference on Machine Learning, pp. 111–118.

[7] Cao, T., Law, M., Fidler, S., 2020. A theoretical analysis of the number of shots in few-shot learning, in: International Conference on Learning Representations.

[8] Chen, L., Zhang, H., Xiao, J., Nie, L., Shao, J., Liu, W., Chua, T.S., 2017. SCA-CNN: Spatial and channel-wise attention in convolutional networks for image captioning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5659–5667.

[9] Chen, M., Fang, Y., Wang, X., Luo, H., Geng, Y., Zhang, X., Huang, C., Liu, W., Wang, B., 2020. Diversity transfer network for few-shot learning., in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 10559–10566.

[10] Chen, W.Y., Liu, Y.C., Kira, Z., Wang, Y.C.F., Huang, J.B., 2018. A closer look at few-shot classification, in: International Conference on Learning Representations.

[11] Chen, Z., Fu, Y., Chen, K., Jiang, Y.G., 2019a. Image block augmentation for one-shot learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 3379–3386.

[12] Chen, Z., Fu, Y., Wang, Y.X., Ma, L., Liu, W., Hebert, M., 2019b. Image deformation meta-networks for one-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8680–8689.

[13] Cheng, M.M., Mitra, N.J., Huang, X., Torr, P.H., Hu, S.M., 2014. Global contrast based salient region detection. IEEE transactions on pattern analysis and machine intelligence 37, 569–582.

[14] Das, A., Agrawal, H., Zitnick, L., Parikh, D., Batra, D., 2017. Human attention in visual question answering: Do humans and deep networks look at the same regions? Computer Vision and Image Understanding 163, 90–100.

[15] Dhillon, G.S., Chaudhari, P., Ravichandran, A., Soatto, S., 2020. A baseline for few-shot image classification, in: International Conference on Learning Representations.

[16] Edwards, H., Storkey, A., 2017. Towards a neural statistician, in: International Conference on Learning Representations.

[17] Fang, H., Gupta, S., Iandola, F., Srivastava, R.K., Deng, L., Dollár, P., Gao, J., He, X., Mitchell, M., Platt, J.C., et al., 2015. From captions

to visual concepts and back, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1473–1482.

[18] Fei-Fei, L., Fergus, R., Perona, P., 2006. One-shot learning of object categories. IEEE Transactions on Pattern Analysis and Machine Intelligence 28, 594–611.

[19] Finn, C., Abbeel, P., Levine, S., 2017. Model-agnostic meta-learning for fast adaptation of deep networks, in: International Conference on Machine Learning, pp. 1126–1135.

[20] Finn, C., Xu, K., Levine, S., 2018. Probabilistic model-agnostic meta-learning, in: Advances in Neural Information Processing Systems, pp. 9516–9527.

[21] Flennerhag, S., Rusu, A.A., Pascanu, R., Visin, F., Yin, H., Hadsell, R., 2020. Meta-learning with warped gradient descent, in: International Conference on Learning Representations.

[22] Gao, H., Shou, Z., Zareian, A., Zhang, H., Chang, S.F., 2018. Low-shot learning via covariance-preserving adversarial augmentation networks, in: Advances in Neural Information Processing Systems, pp. 975–985.

[23] Garcia, V., Bruna, J., 2018. Few-shot learning with graph neural networks, in: International Conference on Learning Representations.

[24] Ghiasi, G., Lin, T.Y., Le, Q.V., 2018. Dropblock: A regularization method for convolutional networks, in: Advances in Neural Information Processing Systems, pp. 10727–10737.

[25] Gidaris, S., Komodakis, N., 2018. Dynamic few-shot visual learning without forgetting, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4367–4375.

[26] Gidaris, S., Komodakis, N., 2019. Generating classification weights with gnn denoising autoencoders for few-shot learning, in: Proceedings of the

IEEE Conference on Computer Vision and Pattern Recognition, pp. 21–30.

[27] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets, in: Advances in Neural Information Processing Systems, pp. 2672–2680.

[28] He, J., Feng, J., Liu, X., Cheng, T., Lin, T.H., Chung, H., Chang, S.F., 2012. Mobile product search with bag of hash bits and boundary reranking, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE. pp. 3005–3012.

[29] He, K., Zhang, X., Ren, S., Sun, J., 2016. Identity mappings in deep residual networks, in: Proceedings of the European Conference on Computer Vision, pp. 630–645.

[30] Hu, J., Shen, L., Sun, G., 2018. Squeeze-and-excitation networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7132–7141.

[31] Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, pp. 448–456.

[32] Jamal, M.A., Qi, G.J., 2019. Task agnostic meta-learning for few-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 11719–11727.

[33] Jetley, S., Lord, N.A., Lee, N., Torr, P.H., 2018. Learn to pay attention, in: International Conference on Learning Representations.

[34] Jia, Y., Huang, C., Darrell, T., 2012. Beyond spatial pyramids: Receptive field learning for pooled image features, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3370–3377.

[35] Jiang, H., Wang, J., Yuan, Z., Wu, Y., Zheng, N., Li, S., 2013. Salient object detection: A discriminative regional feature integration approach, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2083–2090.

[36] Kim, J., Kim, T., Kim, S., Yoo, C.D., 2019. Edge-labeling graph neural network for few-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 11–20.

[37] Kingma, D.P., Ba, J., 2015. ADAM: A method for stochastic optimization, in: International Conference on Learning Representations.

[38] Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, pp. 1097–1105.

[39] Lake, B.M., Salakhutdinov, R., Tenenbaum, J.B., 2015. Human-level concept learning through probabilistic program induction. Science 350, 1332–1338.

[40] Lee, C.Y., Gallagher, P.W., Tu, Z., 2016. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree, in: International Conference on Artificial Intelligence and Statistics, pp. 464–472.

[41] Lee, K., Maji, S., Ravichandran, A., Soatto, S., 2019. Meta-learning with differentiable convex optimization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 10657–10665.

[42] Li, A., Huang, W., Lan, X., Feng, J., Li, Z., Wang, L., 2020a. Boosting few-shot learning with adaptive margin loss, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 12576–12584.

[43] Li, H., Eigen, D., Dodge, S., Zeiler, M., Wang, X., 2019a. Finding task-relevant features for few-shot learning by category traversal, in: Proceed-

ings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–10.

[44] Li, J., Jing, M., Lu, K., Zhu, L., Shen, H.T., 2021. Investigating the bilateral connections in generative zero-shot learning. IEEE Transactions on Cybernetics .

[45] Li, J., Jing, M., Lu, K., Zhu, L., Yang, Y., Huang, Z., 2019b. Alleviating feature confusion for generative zero-shot learning, in: Proceedings of the 27th ACM International Conference on Multimedia, pp. 1587–1595.

[46] Li, J., Jing, M., Zhu, L., Ding, Z., Lu, K., Yang, Y., 2020b. Learning modality-invariant latent representations for generalized zero-shot learning, in: Proceedings of the 28th ACM International Conference on Multimedia, pp. 1348–1356.

[47] Li, K., Zhang, Y., Li, K., Fu, Y., 2020c. Adversarial feature hallucination networks for few-shot learning, in: Proceedings of the IEEE conference on computer vision and pattern recognition.

[48] Li, X., Sun, Q., Liu, Y., Zhou, Q., Zheng, S., Chua, T.S., Schiele, B., 2019c. Learning to self-train for semi-supervised few-shot classification, in: Advances in Neural Information Processing Systems, pp. 10276–10286.

[49] Li, Z., Zhou, F., Chen, F., Li, H., 2017. Meta-SGD: Learning to learn quickly for few shot learning. arXiv preprint arXiv:1707.09835 .

[50] Lifchitz, Y., Avrithis, Y., Picard, S., Bursuc, A., 2019. Dense classification and implanting for few-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 9258–9267.

[51] Liu, L., Zhou, T., Long, G., Jiang, J., Yao, L., Zhang, C., 2019a. Prototype propagation networks (PPN) for weakly-supervised few-shot learning on category graph, in: International Joint Conference on Artificial Intelligence.

[52] Liu, Y., Lee, J., Park, M., Kim, S., Yang, E., Hwang, S.J., Yang, Y., 2019b. Learning to propagate labels: Transductive propagation network for few-shot learning, in: International Conference on Learning Representations.

[53] Meng, L., Zhao, B., Chang, B., Huang, G., Sun, W., Tung, F., Sigal, L., 2019. Interpretable spatio-temporal attention for video action recognition, in: Proceedings of the IEEE International Conference on Computer Vision Workshops.

[54] Mishra, N., Rohaninejad, M., Chen, X., Abbeel, P., 2018. A simple neural attentive meta-learner, in: International Conference on Learning Representations.

[55] Oh, J., Yoo, H., Kim, C., Yun, S.Y., 2021. BOIL: Towards representation change for few-shot learning, in: International Conference on Learning Representations.

[56] Oreshkin, B., López, P.R., Lacoste, A., 2018. TADAM: Task dependent adaptive metric for improved few-shot learning, in: Advances in Neural Information Processing Systems, pp. 721–731.

[57] Petrović, V., Dimitrijević, V., 2015. Focused pooling for image fusion evaluation. Information Fusion 22, 119–126.

[58] Qi, H., Brown, M., Lowe, D.G., 2018. Low-shot learning with imprinted weights, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5822–5830.

[59] Qiao, S., Liu, C., Shen, W., Yuille, A.L., 2018. Few-shot image recognition by predicting parameters from activations, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7229–7238.

[60] Ravi, S., Larochelle, H., 2017. Optimization as a model for few-shot learning, in: International Conference on Learning Representations.

[61] Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J.B., Larochelle, H., Zemel, R.S., 2018. Meta-learning for semi-supervised few-shot classification, in: International Conference on Learning Representations.

[62] Ren, Z., Gao, S., Chia, L.T., Tsang, I.W.H., 2013. Region-based saliency detection and its application in object recognition. IEEE Transactions on Circuits and Systems for Video Technology 24, 769–779.

[63] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al., 2015. Imagenet large scale visual recognition challenge. International Journal of Computer Vision 115, 211–252.

[64] Rusu, A.A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., Hadsell, R., 2019. Meta-learning with latent embedding optimization, in: International Conference on Learning Representations.

[65] Rutishauser, U., Walther, D., Koch, C., Perona, P., 2004. Is bottom-up attention useful for object recognition?, in: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004., IEEE. pp. II–II.

[66] Saeedan, F., Weber, N., Goesele, M., Roth, S., 2018. Detail-preserving pooling in deep networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 9108–9116.

[67] Simon, C., Koniusz, P., Nock, R., Harandi, M., 2020. Adaptive subspaces for few-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4136–4145.

[68] Snell, J., Swersky, K., Zemel, R., 2017. Prototypical networks for few-shot learning, in: Advances in Neural Information Processing Systems, pp. 4077–4087.

[69] Song, H., Torres, M.T., Özcan, E., Triguero, I., 2019. L2AE-D: Learning to aggregate embeddings for few-shot learning with meta-level dropout. arXiv preprint arXiv:1904.04339 .

[70] Sun, Q., Liu, Y., Chua, T.S., Schiele, B., 2019. Meta-transfer learning for few-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 403–412.

[71] Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P.H., Hospedales, T.M., 2018. Learning to compare: Relation network for few-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1199–1208.

[72] Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al., 2016. Matching networks for one shot learning, in: Advances in Neural Information Processing Systems, pp. 3630–3638.

[73] Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S., 2011. The caltech-ucsd birds-200-2011 dataset .

[74] Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X., Tang, X., 2017a. Residual attention network for image classification, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3156–3164.

[75] Wang, L., Lu, H., Wang, Y., Feng, M., Wang, D., Yin, B., Ruan, X., 2017b. Learning to detect salient objects with image-level supervision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 136–145.

[76] Wang, R., Xu, K., Liu, S., Chen, P.Y., Weng, T.W., Gan, C., Wang, M., 2021. On fast adversarial robustness adaptation in model-agnostic meta-learning, in: International Conference on Learning Representations.

[77] Wang, W., Shen, J., Ling, H., 2018a. A deep network solution for attention and aesthetics aware photo cropping. IEEE transactions on pattern analysis and machine intelligence 41, 1531–1544.

[78] Wang, X., You, S., Li, X., Ma, H., 2018b. Weakly-supervised semantic segmentation by iteratively mining common object features, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1354–1362.

[79] Wang, Y., Yao, Q., Kwok, J.T., Ni, L.M., 2020. Generalizing from a few examples: A survey on few-shot learning. ACM Computing Surveys .

[80] Wang, Y.X., Girshick, R., Hebert, M., Hariharan, B., 2018c. Low-shot learning from imaginary data, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7278–7286.

[81] Wei, Y., Liang, X., Chen, Y., Shen, X., Cheng, M.M., Feng, J., Zhao, Y., Yan, S., 2016. Stc: A simple to complex framework for weakly-supervised semantic segmentation. IEEE transactions on pattern analysis and machine intelligence 39, 2314–2320.

[82] Wei, Y., Wen, F., Zhu, W., Sun, J., 2012. Geodesic saliency using background priors, in: European conference on computer vision, Springer. pp. 29–42.

[83] Wertheimer, D., Hariharan, B., 2019. Few-shot learning with localization in realistic settings, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6558–6567.

[84] Woo, S., Park, J., Lee, J.Y., So Kweon, I., 2018. CBAM: Convolutional block attention module, in: Proceedings of the European Conference on Computer Vision, pp. 3–19.

[85] Xing, C., Rostamzadeh, N., Oreshkin, B., Pinheiro, P.O., 2019. Adaptive cross-modal few-shot learning, in: Advances in Neural Information Processing Systems, pp. 4848–4858.

[86] Yan, S., Zhang, S., He, X., et al., 2019. A dual attention network with semantic embedding for few-shot learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 9079–9086.

[87] Yao, H., Wei, Y., Huang, J., Li, Z., 2019. Hierarchically structured meta-learning, in: International Conference on Machine Learning, pp. 7045–7054.

[88] Ye, H.J., Hu, H., Zhan, D.C., Sha, F., 2020. Few-shot learning via embedding adaptation with set-to-set functions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

[89] Yoon, J., Kim, T., Dia, O., Kim, S., Bengio, Y., Ahn, S., 2018. Bayesian model-agnostic meta-learning, in: Advances in Neural Information Processing Systems, pp. 7332–7342.

[90] Zeiler, M.D., Fergus, R., 2013. Stochastic pooling for regularization of deep convolutional neural networks, in: International Conference on Learning Representations.

[91] Zeiler, M.D., Fergus, R., 2014. Visualizing and understanding convolutional networks, in: Proceedings of the European Conference on Computer Vision, pp. 818–833.

[92] Zhang, D., Meng, D., Zhao, L., Han, J., 2017. Bridging saliency detection to weakly supervised object detection based on self-paced curriculum learning. arXiv preprint arXiv:1703.01290 .

[93] Zhang, H., Zhang, J., Koniusz, P., 2019a. Few-shot learning via saliency-guided hallucination of samples, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2770–2779.

[94] Zhang, L., Liu, Z., Zhang, S., Yang, X., Qiao, H., Huang, K., Hussain, A., 2019b. Cross-modality interactive attention network for multispectral pedestrian detection. Information Fusion 50, 20–29.

[95] Zhang, R., Che, T., Ghahramani, Z., Bengio, Y., Song, Y., 2018. MetaGAN: An adversarial approach to few-shot learning, in: Advances in Neural Information Processing Systems, pp. 2365–2374.

[96] Zhao, F., Zhao, J., Yan, S., Feng, J., 2018. Dynamic conditional networks for few-shot learning, in: Proceedings of the European Conference on Computer Vision, pp. 19–35.

[97] Zhao, J.X., Liu, J.J., Fan, D.P., Cao, Y., Yang, J., Cheng, M.M., 2019. EGNet: Edge guidance network for salient object detection, in: Proceedings of the IEEE International Conference on Computer Vision, pp. 8779–8788.

[98] Zhu, H., Ma, W., Li, L., Jiao, L., Yang, S., Hou, B., 2020. A dual–branch attention fusion deep network for multiresolution remote–sensing image classification. Information Fusion 58, 116–131.

[99] Zhu, W., Liang, S., Wei, Y., Sun, J., 2014. Saliency optimization from robust background detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2814–2821.

[100] Zhu, X., Cheng, D., Zhang, Z., Lin, S., Dai, J., 2019. An empirical study of spatial attention mechanisms in deep networks, in: Proceedings of the IEEE International Conference on Computer Vision, pp. 6688–6697.