Feature Extractor Stacking for Cross-domain Few-shot Meta-learning

Hongyu Wang, Eibe Frank, Bernhard Pfahringer, Michael Mayo and Geoffrey Holmes

Department of Computer Science, University of Waikato.

*Corresponding author(s). E-mail(s):
hw168@students.waikato.ac.nz;

Abstract
Cross-domain few-shot meta-learning (CDFSML) addresses learning problems where knowledge needs to be transferred from several source domains into an instance-scarce target domain with an explicitly different distribution. Recently published CDFSML methods generally construct a universal model that combines knowledge of multiple source domains into one backbone feature extractor. This enables efficient inference but necessitates re-computation of the backbone whenever a new source domain is added. Some of these methods are also incompatible with heterogeneous source domain backbone architectures. We propose feature extractor stacking (FES), a new CDFSML method for combining information from a collection of backbones, which can utilise heterogeneous pretrained backbones out of the box, and does not maintain a universal model that needs to be re-computed when its backbone collection is updated. We present the basic FES algorithm, which is inspired by the classic stacking approach to meta-learning, and also introduce two variants: convolutional FES (ConFES) and regularised FES (ReFES). Given a target-domain task, these algorithms fine-tune each backbone independently, use cross-validation to extract meta training data from the support set, and learn a simple linear meta-classifier from this data. We evaluate our FES methods on the well-known Meta-Dataset benchmark, targeting image classification with convolutional neural networks, and show that they can achieve state-of-the-art performance.

Keywords: Cross-domain few-shot learning, Meta-learning, Pretrained feature extractors, Stacking
1 Introduction

Cross-domain few-shot learning (CDFSL) addresses the problem that state-of-the-art machine learning methods, such as convolutional neural networks (CNN) for image classification, generally require a large amount of labelled training data to achieve high predictive accuracy when trained from scratch. CDFSL algorithms are designed for scenarios where only a few labelled training instances are available in the form of a so-called “support set”. The aim is to nevertheless achieve high accuracy when predicting labels for instances of the target domain that have never been seen before, i.e., the so-called “query set”. This can generally only be achieved by applying transfer learning: taking knowledge gleaned from one or several source domains with large-scale training data and using this knowledge to inform learning in a few-shot target domain.

In CDFSL, the source domain(s) and the target domain are assumed to have potentially very distinct properties. This cross-domain setting is arguably more realistic than the “in-domain” scenario, used in some few-shot learning literature (Vinyals, Blundell, Lillicrap, Kavukcuoglu, & Wierstra, 2016), where the source and the target domains comprise mutually exclusive sets of classes obtained from the same dataset. It also yields harder learning problems due to greater domain shift.

Meta-learning algorithms are the top performers in CDFSL benchmarks. In this context, meta-learning algorithms are designed to learn how to most appropriately transfer knowledge from source domains into a target domain model using a support set. Performance of meta-learning is measured by “meta-testing”—evaluating accuracy of the generated model on a corresponding labelled target-domain query set.

Meta-learning is a term used loosely in the CDFSL context, where it refers to learning from multiple domains with a meta-learner. It is used to differentiate from single-domain learning (SDL), as well as vanilla multi-domain learning (MDL) with one feature extractor and multiple classification heads. Meta-learning has been shown to outperform SDL (Triantafillou et al., 2020) and MDL (W. Li, Liu, & Bilen, 2021).

A majority of recently published cross-domain few-shot meta-learning (CDFSML) methods involve building a universal model from a collection of backbones, with each backbone pretrained in a distinct source domain. The universal-model paradigm is generally efficient at meta-test time, in the sense that only a single universal feature extractor is used and fine-tuned on the support set, usually in conjunction with a simple robust classifier that turns extracted feature vectors into predictions. However, training the universal model is computationally expensive, and some methods constrain all backbones to the same architecture as the intended universal model (Triantafillou, Larochelle, Zemel, & Dumoulin, 2021), rendering them inapplicable to heterogeneous backbone collections that are likely to occur in real-world practice. Moreover, they may require heuristic adjustment based on pre-existing

---

1Note that, strictly speaking, this also creates distinct domains because the joint probability distributions will differ. However, they will be strongly related.
domain knowledge to function well. For example, given a source domain/backbone collection for image classification consisting of ImageNet (Deng et al., 2009; Russakovsky et al., 2015), along with other, less comprehensive source domains, authors often—in a seemingly ad hoc manner, assign greater importance to the ImageNet backbone during training (W. Li et al., 2021; Triantafillou et al., 2021). This achieves good performance on benchmarks, which normally include target domains such as CIFAR-10 that are quite similar to ImageNet in nature, but its usefulness in real-world applications can be questioned. Lastly, the process of deriving a universal model is non-incremental, which means it needs to be re-run if a backbone is updated or added, and normally requires access to source domain data, which may not be available.

As an alternative approach that avoids these shortcomings, we propose a novel “lazy” CDFSML method, termed feature extractor stacking (FES), that fine-tunes each backbone independently and trains a meta-classifier using a form of stacked generalisation (Wolpert, 1992) during meta-testing. FES is fully compatible with heterogeneous backbone collections, imposing no constraints on their architecture or fine-tuning configuration. FES assumes equal importance of the backbones a priori and determines their task-specific relevance based purely on the support set. Moreover, FES does not require any source domain data or derivation of a universal model.

Along with the basic FES algorithm, which applies a simple linear meta-classifier and is described in Section 3.1, we present two variants: convolutional FES (ConFES) in Section 3.3 and regularised FES (ReFES) in Section 3.4. ConFES replaces the flat global kernel of FES with a hierarchy of depthwise convolutional kernels, reducing the number of parameters in the meta-classifier. ReFES applies fused lasso regularisation (Tibshirani, Saunders, Rosset, Zhu, & Knight, 2005) to the meta-classifier of FES to reduce the weights of irrelevant snapshots and induce smooth weight transition between adjacent snapshots.

We evaluate FES and its variants on the Meta-Dataset benchmark (Triantafillou et al., 2020), which contains eight source domains and five target domains, and include five additional target domains: CropDisease, EuroSAT, ISIC, ChestX, and Food101 (Bossard, Guillaumin, & Gool, 2014; Guo et al., 2020). We show that FES outperforms three recent universal-model methods: TSA (W. Li, Liu, & Bilen, 2022), URL (W. Li et al., 2021), and FLUTE (Triantafillou et al., 2021), and advances the state of the art on Meta-Dataset. We also discuss practical advantages of FES in real-world scenarios, as FES can work with heterogeneous backbones out of the box and does not need to train a universal model.

2 Related work

As our experiments are based on the Meta-Dataset framework for evaluating CDFSML methods, we review this benchmark first before discussing meta-learning methods that we compare to in our experiments. We also briefly review other noteworthy methods in the literature.
2.1 Benchmark

The Meta-Dataset (Triantafillou et al., 2020) benchmark has multiple configurations; we describe the CDFSML configuration that we use, as do most recent publications in the field. It contains eight source domains: ILSVRC-2012 (ImageNet), Omniglot, Aircraft, CUB-200-2011 (Birds), Describable Textures, Quick Draw, Fungi, and VGG Flower. Recent work utilising Meta-Dataset (Requeima, Gordon, Bronskill, Nowozin, & Turner, 2019) has extended its original set of two target domains, Traffic Signs and MSCOCO, by adding three more: MNIST, CIFAR10, and CIFAR100. For an even more comprehensive evaluation, we add four target domains from the CDFSL benchmark in Guo et al. (2020)—CropDisease, EuroSAT, ISIC, and ChestX—but additionally also employ Food101 (Bossard et al., 2014). Only the 250 sanitised test images in each Food101 class are used in our experiments.

The Meta-Dataset framework splits each source domain into three partitions: training, validation, and test. The partitions are mutually exclusive in terms of their classes, with the training partition containing approximately 70% of source domain classes and the validation and test partitions containing approximately 15% each. The training and validation partitions are made available to the CDFSML method for “meta-training”, where the training partition is generally used to train backbone models and the validation split to aid hyperparameter tuning. The test partition is reserved for evaluating the CDFSML method by sampling few-shot episodes (i.e., meta-testing); the term “episode” refers to the process of sampling a support set and a query set, training a classifier on the support set, and evaluating it on the query set.

In contrast, the entire target domain data can be used for sampling episodes to evaluate few-shot learning in these domains. It is important to note that, by definition, only tasks sampled from target domains truly measure CDFSL performance. Using terminology that is common in this context, good performance in these domains indicates “strong generalisation”; good performance on tasks sampled from source domain test partitions indicates “weak generalisation”.

The most commonly used method to evaluate CDFSML algorithms on Meta-Dataset is to generate 600 any-way any-shot episodes from each dataset, and measure each algorithm’s mean accuracy on these 600 episodes, as well as the 95% confidence interval. Episode generation involves the target domains for strong generalisation and test partitions of the source domains for weak generalisation. Any-way any-shot sampling means the number of classes for each episode and the number of support instances per class are arbitrary, leading to imbalanced support sets more representative of real-world scenarios than fixed-way fixed-shot episodes. The query set is balanced in the Meta-Dataset setting. We adhere to this evaluation method in our experiments.

2.2 Methods included in the experimental comparison

Two recently published CDFSML methods that advanced the state-of-the-art on Meta-Dataset are Few-shot Learning with a Universal TEmplate
(FLUTE) (Triantafillou et al., 2021) and Universal Representation Learning (URL) (W. Li et al., 2021). Even more recently, based on a URL universal model, a fine-tuning method using Task-Specific Adaptors (TSA) (W. Li et al., 2022) improved accuracy on some target domains even further. We compare our new FES approach to these methods in our experiments.

2.2.1 Few-shot Learning with a Universal TEmplate.

Based on the FiLM approach (Perez, Strub, de Vries, Dumoulin, & Courville, 2018), FLUTE trains a universal model in the source domains, employing the ResNet18 architecture (He, Zhang, Ren, & Sun, 2016) widely used in CDFSML, but maintaining a separate set of batch normalisation (Ioffe & Szegedy, 2015) parameters for each domain. The ResNet “template” contains one set of convolutional weights shared across all source domains, and only the batch normalisation parameters are specific to each source domain. FLUTE jointly trains the template in all source domains. At each training iteration, a random source domain is selected—with ImageNet having a 50% probability of being selected and the other seven source domains evenly sharing the other 50% probability—and a batch of input data is sampled from the selected source domain. In forward propagation, the input batch flows through the shared convolutional layers and the selected domain’s set of batch normalisation layers, and loss is computed by applying a cosine classifier (W. Chen, Liu, Kira, Wang, & Huang, 2019; Y. Chen, Liu, Xu, Darrell, & Wang, 2021). A nuance of FLUTE training is that backpropagation is performed using a “meta-batch” of eight individual batches: the intention is to stabilise training by aggregating loss values across multiple domains. Hyperparameter tuning is performed using episodes sampled from source domain validation partitions.

When the template is trained, snapshots are frequently saved. The final template is chosen as the snapshot that yields the best performance on the source domains’ validation partitions. To establish performance, few-shot episodes are sampled from these partitions. For each episode, feature vectors are extracted using the shared convolutional layers and the domain’s set of batch normalisation layers. Accuracy is computed using a nearest-centroid classifier (Mensink, Verbeek, Perronnin, & Csurka, 2013; Snell, Swersky, & Zemel, 2017).

One more component of FLUTE, produced in a separate meta-training phase, is a blender network, which is a dataset classifier based on a permutation-invariant set encoder (Zaheer et al., 2017) followed by a linear layer. Given a batch of instances, the blender predicts, as a probability distribution, the source domain from which the batch is sampled. It is trained on batches sampled from the source domains’ training partitions, and the final blender model is chosen using batches from the validation partitions.

Given a few-shot episode at meta-test time, the blender uses the support set to produce a probability distribution. These probabilities in turn are used to form a linear combination of the source-domain-specific batch normalisation weights. Along with the shared convolutional weights from the template,
this forms the initial set of parameters for the ResNet18 feature extractor, which is applied in conjunction with a nearest-centroid classifier. The model’s batch normalisation parameters are then fine-tuned on the support set while its convolutional weights remain fixed.

2.2.2 Universal Representation Learning

The URL algorithm also generates a universal model. It first pretrains domain-specific ResNet18 backbones independently. Then, a separate ResNet18 feature extractor is trained to form a universal model by distillation. This is achieved by making this model learn to match each backbone’s output feature vectors and logits using instances sampled from the backbone’s corresponding domain. To this end, the universal model contains pairs of auxiliary domain-specific components that each comprise 1) a projection layer that transforms the universal extractor’s feature vectors to match those of each domain-specific backbone, and 2) a classifier layer trained to match the logits produced by each backbone.

In the experiments in W. Li et al. (2021), ImageNet is made more prominent in distillation, as ImageNet instances make up 50% of each mini-batch and the other seven source domains evenly make up the rest. Snapshots of the universal feature extractor are saved at predefined intervals during knowledge distillation. Episodes sampled from source domain validation partitions are used to select the best snapshot as a form of early stopping.

After meta-training, the auxiliary components of the universal model are discarded, leaving only the feature extractor. At meta-test time, this extractor is frozen, and a projection layer is initialised with an identity weight matrix and trained using the support set. The projected feature vectors are used to build a nearest-centroid classifier, and cosine similarity values between a feature vector and the centroids are used as its logits. Fine-tuning minimises the cross-entropy loss computed using the support set’s classification by the nearest-centroid classifier and the support labels. Note that during fine-tuning, as the projection layer is optimised, projected support feature vectors change, and their centroids change as well. The fine-tuning effect can be interpreted as forming better clusters with projected support feature vectors.

2.2.3 Task-Specific Adaptors

TSA (W. Li et al., 2022) is a fine-tuning method suitable for CDFSML. Given a pretrained backbone, trainable task-specific adaptors are attached to it, and the support set is used to optimise the adaptors with the backbone’s original weights frozen. Like URL, TSA also attaches a trainable linear projection layer and a robust classifier to the end of the feature extractor during fine-tuning, but it adds further components. Among multiple configurations examined, the most effective approach for few-shot image classification found in W. Li et al. (2022) is to attach channel projection matrices as residual connections to a model’s convolutional layers. In W. Li et al. (2022), TSA was used in
conjunction with a URL-distilled universal backbone, but TSA can practically be applied to any CNN architecture.

2.3 Other work on CDFSML

We review additional noteworthy CDFSML methods here. These methods preceed FLUTE, URL, and TSA chronologically and achieve lower accuracy than results presented in W. Li et al. (2021, 2022); Triantafillou et al. (2021). Hence, in the experiments presented in this paper, we only compare to FLUTE, URL, and TSA.

2.3.1 Selecting Relevant Features from a Universal Representation

SUR (Dvornik, Schmid, & Mairal, 2020) is a CDFSML method that utilises independently pretrained feature extractor backbones directly for meta-testing. Each backbone is used to extract a set of feature vectors from the support set, with a trainable weight assigned to it. Feature vectors are multiplied by their respective weights and concatenated to provide input to a nearest-centroid classifier. The weights are trained by optimising loss of the classifier on the support set. SUR is similar to URL in the meta-testing phase, as both make predictions with a nearest-centroid classifier and optimise parameters on the support set; the primary difference is that URL maintains a universal model while SUR uses the backbones directly.

2.3.2 Universal Representation Transformer

URT (L. Liu, Hamilton, Long, Jiang, & Larochelle, 2021) also assigns a weight to each source domain backbone during meta-testing. However, it utilises a weight assignment model learned using meta-training instead of direct optimisation on the support set to obtain the weights. To this end, URT trains an attention mechanism (Vaswani et al., 2017) that learns to assign appropriate weights to source domain feature extractors given a few-shot episode. The weight assignment model is trained and has its hyperparameters selected using episodes sampled from the source domains’ training and validation partitions.

2.3.3 Conditional Neural Adaptive Processes

The CNAPs method, as proposed in Requeima et al. (2019), uses a backbone pretrained in a large source domain, e.g., ImageNet (Deng et al., 2009; Russakovsky et al., 2015), and meta-trains adaptation networks, using episodes sampled from the source domains, to produce task-specific FiLM (Perez et al., 2018) transformations and a linear classifier for each few-shot episode.

A variant, Simple CNAPs (Bateni, Goyal, Masrani, Wood, & Sigal, 2020), was later proposed utilising a non-parametric Mahalanobis distance (Galeano, Joseph, & Lillo, 2015) measure in place of the classifier adaptation network of CNAPs, reducing the parameter count and improving CDFSML
performance. A transductive version of Simple CNAPs was subsequently also proposed (Bateni, Barber, van de Meent, & Wood, 2022), making use of clustering of query instances in feature space to achieve better performance than Simple CNAPs, assuming that the query set is available as a batch instead of a sequential stream of incoming instances. As most other CDFSML methods do not rely on such an assumption, they cannot be compared to transductive CNAPs on an even footing.

2.3.4 Multi-Mode Modulator

Tri-M (Y. Liu et al., 2021), akin to CNAPs, uses a backbone pretrained in a large-scale source domain, and meta-trains a modulation network using source domain episodes to generate appropriate FiLM transformations for each few-shot episode. Tri-M maintains two sets of transformations—a domain-specific one and a domain-cooperative one—and its resulting FiLM transformation is a combination of the two. Tri-M determines a source domain for its domain-specific transformation in a way similar to how FLUTE (Triantafillou et al., 2021) utilises its blender network and uses an attention mechanism (Vaswani et al., 2017) to compose its domain-cooperative transformation from relevant source domains.

3 Cross-domain Few-shot Meta-Learning using Stacking

Considering the CDFSML methods discussed in the previous section, the SUR method stands out because it does not require meta-training other than pretraining individual backbones, and performs “lazy” learning in the sense that significant work is only performed once the support set for a few-shot episode becomes available. This makes it very flexible because new backbones can be added at any time. However, SUR does not yield state-of-the-art performance. The new methods presented in this paper are inspired by SUR and the old and established method of applying stacked generalisation to learning a meta-classifier. There are four primary differences between SUR and our stacking-based methods: 1) the source domain backbones are fined-tuned on the support set to extract more information from this data, 2) two-fold cross-validation is used to generate training data for the meta-classifier to tackle overfitting, 3) the feature vectors of this training data consist of logits obtained from classifier layers attached to the backbones for fine-tuning, and 4) multiple snapshots of each backbone are stored during fine-tuning and used to obtain sets of logits, adding further richness to the data available for training the meta-classifier.

In the following, we first explain the basic method of feature extractor stacking (FES) in detail and prove convexity of its optimisation, before describing two variants: convolutional FES (ConFES) and regularised FES (ReFES).
3.1 Feature Extractor Stacking

FES is formulated into three phases: fine-tuning backbones to obtain snapshots, cross-validation to produce meta-training data for a meta-classifier, and training of the meta-classifier. Figure 1 depicts the FES framework.

3.1.1 Fine-tuning the backbones

We use \( f_{\Phi_1}, f_{\Phi_2}, \ldots, f_{\Phi_K} \) (or just \( \Phi_1, \Phi_2, \ldots, \Phi_K \) for brevity) to denote the collection of pretrained feature extractors, where \( \Phi \) represents the corresponding extractor’s parameters and \( K \) is the number of source domains. The support set of a few-shot episode is denoted \( S \) and the query set \( Q \). \( S \) contains \( N \) instances belonging to \( C \) classes. We fine-tune each backbone independently on \( S \). As \( f_\Phi \) is a feature extractor, a classifier \( g \) with parameters \( \Theta_1 \) is attached to \( f_\Phi \) to produce logits. Auxiliary components with parameters \( \Theta_2 \) may also be introduced to the model to aid fine-tuning, such as with TSA (W. Li et al., 2022). The resulting model is defined as \( h_\Psi = g_{\Theta_1} \circ f_{(\Phi, \Theta_2)} \), where we use \( \Psi \) to denote the combination of all parameters. It is possible for \( \Theta_2 \) to be \( \emptyset \), as auxiliary fine-tuning components are optional. \( J \) snapshots are saved sequentially at different fine-tuning iterations of \( h_\Psi \). Each snapshot contains parameters \( \Psi_k[S] \), where \( k \in [1, K] \) and \( j \in [1, J] \), with \( S \) denoting the fine-tuning set used.

3.1.2 Cross-validation to obtain meta-level training data

In stacked generalisation, cross-validation is employed to obtain training data for the meta-classifier to combat overfitting, and it is applied in FES as well. More specifically, we apply stratified two-fold cross-validation to the support set \( S \), producing two splits \( S_1 \) and \( S_2 \), which will take turns serving as the training split \( S_{\text{train}} \) and the test split \( S_{\text{test}} \). It is possible to employ more folds in FES, but using additional folds did not yield performance gains in our experiments.

Training on one of the training splits amounts to fine-tuning a network \( h_\Psi \) on this data. In principle, this could be done for a fixed number of iterations, and once complete, logits on the corresponding test split could be obtained as training data for the meta-classifier. However, this naive approach does not work well because it is not known how many iterations should be performed for fine-tuning to maximise accuracy of the full learning system. The approach we propose and evaluate in this paper is instead based on the idea that we can take multiple snapshots of the models during fine-tuning and use all the snapshots’ logits on the test folds for training the meta-classifier. In other words, the learning algorithm for the meta-classifier will be responsible for deciding which backbone snapshots are the most useful ones for making accurate predictions on the test folds.

More specifically, given a pair \((S_{\text{train}}, S_{\text{test}})\) and a backbone \( h_\Psi \), we fine-tune \( h_\Psi \) on \( S_{\text{train}} \) with the same configuration used to obtain \( h_\Psi[S] \), e.g.,
Fig. 1: Framework of FES. Given a backbone collection with $K$ backbones, each backbone $\Phi$ is set up as a network $\Psi$ for fine-tuning. The support set $S$ is split into $S_1$ and $S_2$ using stratified cross-validation. Each network $\Psi$ is fine-tuned on one split, producing $J$ snapshots in the process, and these snapshots are used to extract logits from the other split. Logits extracted from both splits are combined into cross-validated logits of the full support set, which are used to train a meta-classifier $W$ to fit $S$’s labels. The full $S$ is then used to fine-tune $\Psi$, producing snapshots with which query set $Q$’s logits are extracted at test time. $W$ takes $Q$’s logits as input and predicts $Q$’s labels.
Fig. 2: FES uses a global kernel to compute meta-logits from the snapshots’ base logits. The global kernel is essentially flat since it makes no use of the snapshots’ temporal relations. For demonstration purposes, this figure and the following ones assume three backbones \((K = 3)\), five fine-tuning snapshots per backbone \((J = 5)\), and a two-class problem \((C = 2)\).

optimiser, learning rate, etc., and save snapshots \(h_{Ψ_j[S\text{train}]}\) at the same iterations as \(h_{Ψ_j[S]}\). Logits \(L_j[S\text{test}]\) are extracted from \(S\text{test}\) with each \(h_{Ψ_j[S\text{train}]}\), i.e., \(L_j[S\text{test}] = h_{Ψ_j[S\text{train}]}(S\text{test})\). Using this approach, the two splits \(S_1\) and \(S_2\) can be used to alternately fine-tune backbones and produce logits \(L_j[S_1]\) and \(L_j[S_2]\), which are combined into \(L_j[CV]\), i.e., logits for every support set instance extracted using cross-validation. For all \(K\) backbones, \(L_j^K[CV]\) is a matrix of shape \(N × K × J × C\), i.e., \(N\) support instances converted into logits on \(C\) classes extracted by \(K × J\) snapshot models, ready to serve as meta-learning instances: FES prevents overfitting in the same way as the classic stacking method for ensemble learning, by using cross-validated data for training the meta-classifier.

3.1.3 Meta-classifier training

The FES meta-classifier is a weight matrix \(W\) of shape \(K × J\), with \(W_k^j\) representing \(Ψ_k^j\)’s weight. Given a meta-learning instance \(l\) of shape \(K × J × C\), the meta-classifier’s output logits \(l^W\) are obtained using a simple weighted average:

\[
l^W[c] = \sum_{k=1}^{K} \sum_{j=1}^{J} W_k^j \cdot l_k^j[c],
\]

where \(c\) is one of the \(C\) classes. We compute the cross-entropy loss using the \(N\) support set meta-classifier outputs \(L^W\) and the one-hot-encoded labels \(Y\), i.e., \(-\sum_{n=1}^{N} Y_n \log(\text{softmax}(L_n^W))\), which we minimise by training \(W\). For interpretability, we constrain all values in \(W\) to be non-negative by clipping negative weights with ReLU. The FES meta-classifier is shown in Figure 2.
After training, $W$ is used with Equation 1 to compute meta-logits for the query set $Q$ using the logits $L^J_K[Q]$ computed by the saved snapshots $\Psi^J_K[S]$. Then, a softmax function is used to obtain class probability estimates.

### 3.2 Proof of Convexity

Given a meta-learning instance $l$, consisting of logits obtained from the backbone models, which the meta-classifier transforms into meta-level logits $l^W$, and the label $c_y$, the negative log-likelihood loss $\ell$ associated with the meta-classifier’s parameters $W$ is

$$\ell(W) = \log\left(\sum_{i=1}^C e^{l^W[c_i]}\right) - l^W[c_y].$$  \hfill (2)

To prove that optimising FES is a convex problem, we show that for any two values of $W$, named $A$ and $B$, a linear combination of the loss on $A$ and the loss on $B$ is never smaller than the loss obtained for the corresponding linear combination of the parameter values $A$ and $B$, i.e.,

$$\ell(\lambda A + (1 - \lambda)B) \leq \lambda \ell(A) + (1 - \lambda)\ell(B), \lambda \in [0, 1].$$ \hfill (3)

Applying Equation 2 to Equation 3, we get

$$\log\left(\sum_{i=1}^C e^{l^{(\lambda A + (1 - \lambda)B)[c_i]}}\right) - l^{(\lambda A + (1 - \lambda)B)[c_y]} \leq \lambda(\log\left(\sum_{i=1}^C e^{l^{A}[c_i]}\right) - l^A[c_y]) + (1 - \lambda)(\log\left(\sum_{i=1}^C e^{l^{B}[c_i]}\right) - l^B[c_y]),$$

which can be simplified into

$$\log\left(\sum_{i=1}^C e^{l^{(\lambda A + (1 - \lambda)B)[c_i]}}\right) \leq \lambda \log\left(\sum_{i=1}^C e^{l^{A}[c_i]}\right) + (1 - \lambda) \log\left(\sum_{i=1}^C e^{l^{B}[c_i]}\right),$$ \hfill (4)

because using Equation 1, we have

$$l^{(\lambda A + (1 - \lambda)B)[c_y]}$$

$$= \sum_{k=1}^K \sum_{j=1}^J (\lambda A^j_k + (1 - \lambda)B^j_k) \cdot l^j_k[c_y]$$

$$= \sum_{k=1}^K \sum_{j=1}^J \lambda A^j_k \cdot l^j_k[c_y] + \sum_{k=1}^K \sum_{j=1}^J (1 - \lambda)B^j_k \cdot l^j_k[c_y]$$

$$= \sum_{k=1}^K \sum_{j=1}^J \lambda A^j_k \cdot l^j_k[c_y] + \lambda \sum_{k=1}^K \sum_{j=1}^J B^j_k \cdot l^j_k[c_y] + (1 - \lambda) \sum_{k=1}^K \sum_{j=1}^J B^j_k \cdot l^j_k[c_y].$$
\[
= \lambda \sum_{k=1}^{K} \sum_{j=1}^{J} A_k^j \cdot t_k^j [c_y] + (1 - \lambda) \sum_{k=1}^{K} \sum_{j=1}^{J} B_k^j \cdot t_k^j [c_y] \\
= \lambda l^A [c_y] + (1 - \lambda) l^B [c_y].
\]

Similarly, Equation 4 can be transformed using Equation 1 into

\[
\log \left( \sum_{i=1}^{C} e^{\lambda l^A[c_i] + (1 - \lambda) l^B[c_i]} \right) \leq \lambda \log \left( \sum_{i=1}^{C} e^{l^A[c_i]} \right) + (1 - \lambda) \log \left( \sum_{i=1}^{C} e^{l^B[c_i]} \right).
\]

It is known that the LogSumExp function \( LSE(x) = \log \left( \sum_{i=1}^{n} e^{x_i} \right) \) is convex. Therefore, we have

\[
\forall n \in \mathbb{Z}^+, \alpha, \beta \in \mathbb{R}^n : LSE(\lambda \alpha + (1 - \lambda) \beta) \leq \lambda LSE(\alpha) + (1 - \lambda) LSE(\beta).
\]

3.3 Convolutional Feature Extractor Stacking

The basic FES approach does not exploit the temporal relation between logits obtained from adjacent snapshots produced during fine-tuning. Convolutional FES (ConFES) replaces the global kernel of FES with a kernel hierarchy, as shown in Figure 3, to treat the collection of logits as a time series. The hierarchy comprises one or more lower-level one-dimensional depthwise convolutional kernels and a top-level global kernel. The depthwise kernels condense the logit output sequence from each backbone’s snapshots into a 1D feature map, while keeping the backbones separate, and the global kernel summarises the feature maps produced by the lower-level kernels.

ConFES is motivated by the assumption that when each backbone is fine-tuned on the support set, it undergoes gradual changes between iterations, and the logits output by sequentially saved snapshots can be considered a time series. Therefore, 1D convolutions can be used to discern informative patterns in the time series data and compute feature maps, which are smaller in size than the raw logit time series, and therefore require fewer parameters in the global kernel than standard FES.
Fig. 3: ConFES replaces the flat kernel of FES with a two-level kernel hierarchy. The base-level kernel is a one-dimensional depthwise, i.e., feature-extractor-wise, convolutional kernel, with predefined kernel and stride sizes. The high-level kernel is global like the one in FES, however, it is applied to the output of the base-level kernel, which requires substantially fewer parameters.

Given $K$ backbones and $J$ snapshots for each backbone, FES requires $K \times J$ parameters. Assuming a two-level ConFES hierarchy, with a base-level convolutional kernel of size $J_b$ and stride $T$, the feature map for each backbone will be of length $J_m = \frac{J - J_b}{T} + 1$, leading to a global kernel size of $K \times (\frac{J - J_b}{T} + 1)$. Including the $K \times J_b$ parameters in the convolutional kernel, ConFES contains $K \times (\frac{J - J_b}{T} + 1 + J_b)$ parameters. In practice, it can generally be assumed that $J \gg J_b \geq T \gg 1$ in order to cover all snapshots with significantly fewer parameters than FES.

ConFES utilises the sequential relation of each backbone’s snapshots through its lower-level 1D depthwise convolutional layers and exhibits substantially fewer parameters than FES, making it less prone to overfitting. Note that Figure 3 is simplified for demonstration purposes and does not reflect well that ConFES maintains fewer parameters; for a practical example of ConFES kernels, please refer to Figure 12.

3.4 Regularised Feature Extractor Stacking

To combat overfitting, an alternative to reducing the number of parameters is to perform regularisation. Regularised FES (ReFES) introduces fused lasso regularisation (Tibshirani et al., 2005) to the meta-classifier used in FES, as shown in Figure 4. Non-zero weights are penalised with a strength of $\lambda_1$, and each feature-extractor-wise weight sequence is smoothed with a strength of $\lambda_2$. The loss is a combination of cross-entropy loss and depthwise fused lasso loss, as formulated in Equation 7, given $K$ backbones, $J$ snapshots per backbone,
Fig. 4: ReFES uses the same global kernel as FES and applies fused lasso regularisation to the kernel’s training process. Fused lasso drives each individual weight towards zero with a regularisation strength of $\lambda_1$ and applies depthwise smoothing to the weight matrix by penalising the weight difference between adjacent snapshots with a regularisation strength of $\lambda_2$.

and a 2D global kernel $W$ of shape $K \times J$.

$$\ell = \ell_{\text{cross-entropy}} + \lambda_1 \sum_{k=1}^{K} \sum_{j=1}^{J} \|W_{jk}\| + \lambda_2 \sum_{k=1}^{K} \sum_{j=1}^{J-1} \|W_{jk} - W_{j+1,k}\|. \quad (7)$$

In addition to encouraging sparse weights like standard lasso, fused lasso also encourages smaller differences between adjacent weights (Tibshirani et al., 2005). Each backbone’s snapshots are ordered by their fine-tuning iterations, and adjacent backbones are likely to be similar. By applying fused lasso regularisation, differences between adjacent weights are penalised, and weight sequences are smoothed.

The stratified two-fold splits $S_1$ and $S_2$ can be used to select appropriate $\lambda_1$ and $\lambda_2$ values for a few-shot episode. In the spirit of grid search with cross-validation, a ReFES meta-classifier is trained on the logits of one split, e.g., $L^{J}_{K}[S_1]$, and tested on the logits of the other split, e.g., $L^{J}_{K}[S_2]$. Different values for $\lambda_1$ and $\lambda_2$ can be explored and the best configuration selected based on the combined accuracy on the two folds. This configuration is then used to train a newly initialised ReFES meta-classifier on the full set of cross-validation logits $L^{J}_{K}[CV]$, and this meta-classifier is used to label the query set instances $Q$ based on their logits $L^{J}_{K}[Q]$. 
3.5 Handling single-instance classes

Meta-Dataset’s sampling scheme (Triantafillou et al., 2020) sometimes produces support sets containing single-instance classes. During cross-validation, single-instance classes need to be removed because if a class’ only instance is in the test split \( S^\text{test} \), then the training split \( S^\text{train} \) will have no instance of that class. FES and its variants can train their meta-classifiers on a subset of the support classes \( C_{\text{sub}} \leq C \), because their kernels only encode the weights of the snapshots, and are inherently myopic to the number of classes \( C \). In Figures 2, 3, and 4, \( C \) can simply be replaced by \( C_{\text{sub}} \) during training.

Given a strict one-shot problem, where all classes exhibit exactly one instance, FES cross-validation is infeasible, as all classes need to be removed during cross-validation, leading to \( L^\text{CV}_{K} = \emptyset \). Therefore, support logits obtained from ordinary fine-tuning need to be used in place of cross-validation logits, i.e., \( L^\text{S}_{K} \) is used to train the FES meta-classifier \( W \) instead of using \( L^\text{CV}_{K} \).

4 Experimental setup

To evaluate FES and its variants on the Meta-Dataset benchmark described in Section 2.1, we use a backbone collection containing eight backbones, each independently pretrained on a Meta-Dataset source domain. In our primary set of experiments, all backbones are ResNet18 models (He et al., 2016) and identical to the source domain backbones used in the publication introducing URL (W. Li et al., 2021).

FES is compatible with any fine-tuning algorithm that is applicable to its individual backbones. In our experiments, we save a snapshot of each backbone before fine-tuning and save a snapshot after each iteration. We evaluate FES with three fine-tuning methods used by state-of-the-art CDFSML methods in the literature:

- **TSA** (W. Li et al., 2022)—matrix residual adaptors attached to convolutional layers, and a fully-connected layer to project feature vectors.
- **URL** (W. Li et al., 2021)—only a fully-connected layer to project feature vectors.
- **FLUTE** (Triantafillou et al., 2021)—scaling and shift factors of batch normalisation layers.

When performing each fine-tuning method for FES, we use the hyperparameters as stated in the source publications, including optimiser type, learning rate, number of iterations, etc., and we compare FES to each source method. The URL (W. Li et al., 2021) and TSA (W. Li et al., 2022) papers fine-tune their feature extractors for 40 iterations, leading to 41 FES snapshots per backbone when using their fine-tuning methods. The FLUTE (Triantafillou et al., 2021) paper fine-tunes its feature extractor for 6 iterations, leading to 7 FES snapshots per backbone when using FLUTE’s fine-tuning method.
Table 1: Meta-Dataset episode statistics.

| Datasets       | support size | class count | mean shot |
|----------------|--------------|-------------|-----------|
|                | min | mean | max | min | mean | max | min | mean | max |
| ilsvrc_2012    | 8   | 380.28 | 498 | 6   | 15.13 | 50 | 1   | 33.53 | 83.00 |
| omniglot       | 5   | 94.38  | 378 | 5   | 19.11  | 47 | 1   | 4.86  | 9.20  |
| aircraft       | 5   | 333.92 | 497 | 5   | 10.04  | 15 | 1   | 35.05 | 76.60 |
| cu_birds       | 8   | 318.22 | 494 | 5   | 17.46  | 30 | 1   | 19.69 | 46.20 |
| dtd            | 5   | 290.94 | 498 | 5   | 6.00   | 7  | 1   | 48.64 | 97.20 |
| quickdraw      | 9   | 410.91 | 497 | 5   | 27.47  | 50 | 1   | 19.83 | 84.60 |
| fungi          | 6   | 350.79 | 494 | 5   | 26.38  | 50 | 1   | 16.31 | 65.43 |
| vgg_flower     | 7   | 290.72 | 497 | 5   | 10.55  | 16 | 1   | 28.36 | 73.80 |
| traffic_sign   | 11  | 416.66 | 497 | 5   | 24.54  | 43 | 1   | 22.03 | 98.40 |
| mscoco         | 9   | 418.97 | 498 | 5   | 23.07  | 40 | 1   | 23.10 | 96.60 |
| mnist          | 5   | 325.46 | 498 | 5   | 7.52   | 10 | 1   | 44.33 | 99.60 |
| cifar10        | 7   | 318.78 | 498 | 5   | 7.47   | 10 | 1   | 44.16 | 99.40 |
| cifar100       | 9   | 409.28 | 497 | 5   | 27.32  | 50 | 1   | 19.74 | 84.60 |
| CropDisease    | 8   | 425.02 | 498 | 5   | 21.77  | 38 | 1   | 24.30 | 95.40 |
| EuroSAT        | 5   | 332.65 | 498 | 5   | 7.54   | 10 | 1   | 45.39 | 99.60 |
| ISIC           | 5   | 282.67 | 498 | 5   | 6.01   | 7  | 1   | 47.45 | 99.60 |
| ChestX         | 5   | 280.74 | 498 | 5   | 5.91   | 7  | 1   | 47.54 | 99.40 |
| Food101        | 7   | 420.13 | 498 | 5   | 26.99  | 50 | 1   | 20.53 | 98.40 |

We adhere to the TSA, URL, and FLUTE papers when replicating and evaluating their methods as benchmarks. Pretrained universal backbones are obtained from the official repositories, and hyperparameter settings are consistent with the papers’ specifications. Note that both the URL and TSA papers used the same URL-distilled universal backbone, and their difference is in fine-tuning, i.e., only fine-tuning a feature projection (URL) or additionally fine-tuning convolutional channel projections (TSA).

We use an LBFGS optimiser to train the meta-classifier, applying its default hyperparameters in the PyTorch library (Paszke et al., 2019), except that we utilise its line search function. A ridge regularisation of strength $10^{-3}$ is applied to FES and ConFES to make the LBFGS optimiser more numerically stable. Adjusting the regularisation strength up or down by an order of magnitude does not substantially affect classification accuracy.

Meta-Dataset’s sampling randomness may cause one or two percent accuracy fluctuation of evaluated methods between different runs, as also stated in URL and TSA’s code repositories (W.-h. Li, Liu, & Bilen, 2022). This fluctuation may exceed the 95% confidence interval of most results, so to eliminate it, we sample 600 episodes from each domain once in Meta-Dataset. The sampled episodes are cached and then used to evaluate all CDFSML methods. In a dataset, the numbers of classes and instances are randomly sampled for each episode, which means that different episodes can contain different numbers of classes and instances. In an episode, the number of instances is randomly sampled for each class, which means that different classes can contain different numbers of instances, and episodes can be class-imbalanced. Triantafillou et al. (2021) pointed out that Meta-Dataset instances need to be shuffled during
sampling in case of datasets with particular ordering, e.g., traffic_sign contains consecutive frames from the same video. But their shuffling solution was implemented as a moving window of size 1000 for streams of instances of each class, which we found to be potentially insufficient, leading to approximately 1% better accuracy in mscoco and 3% better accuracy in ChestX than true random sampling. We found that a window size of 10000 yielded virtually the same level of accuracy as true random sampling, but nevertheless use true random sampling in our experiments, i.e., instances in each class are fully randomised and have equal chance of being selected, and episodes are completely independent of each other. Statistics of our sampling run are shown in Table 1. This also allows us to perform a paired $t$-test on a per-dataset basis as a more sensitive statistical difference test than simply comparing two algorithms’ mean accuracy and confidence intervals. In addition, we rank the algorithms and show their critical difference diagrams (Demsar, 2006) in weak and strong generalisation.

Considering the complexity of the optimisation problem when learning the meta-classifier, it is worth noting that the FES and ReFES meta-classifiers each maintain $8 \times 41 = 328$ parameters if the backbones are fine-tuned for 40 iterations, and $8 \times 7 = 56$ parameters if the backbones are fine-tuned for 6 iterations.

ConFES is applied with a two-level hierarchy, i.e., a low-level depthwise 1D convolutional kernel and a high-level global kernel. For 40-iteration fine-tuning, the convolutional kernel has size $L = 9$ with stride $T = 4$, leading to a feature sequence/global kernel of length 9. Consequently, ConFES has $8 \times 9 + 8 \times 9 = 144$ parameters in total. For 6-iteration fine-tuning, the convolutional kernel has size 3 with stride 2, leading to a global kernel of length 3, and therefore ConFES contains $8 \times 3 + 8 \times 3 = 48$ parameters in total. All parameters are initialised with a constant $(1e^{-3})^\frac{1}{h}$, where $h$ is the number of hierarchical levels in the meta-classifier. Therefore, FES and ReFES are initialised with $1e^{-3}$, and a two-level ConFES hierarchy is initialised with $(1e^{-3})^\frac{1}{2}$. This initialisation is deterministic and ensures that the product of weights from all levels is close to $1e^{-3}$, which is small enough for optimisation to go in either direction, but also big enough to avoid exceedingly small derivatives in gradient-based optimisers.

To facilitate grid search for the $\lambda_1$ and $\lambda_2$ values of ReFES, a pool of eight potential values is provided for each hyperparameter: 1, $1e^{-1}$, $1e^{-2}$, $1e^{-3}$, $1e^{-4}$, $1e^{-5}$, $1e^{-6}$, and 0.

5 Results

We present CDFSML results of FES, ConFES, ReFES, and the competing methods TSA, URL, and FLUTE, on the Meta-Dataset benchmark and show that FES and its variants advance the state of the art on this benchmark. We then visually analyse an example of trained FES, ConFES, and ReFES kernels. Lastly, we examine the ability of FES, ConFES, and ReFES to omit snapshots with their non-negative kernels.
### Table 2: Meta-Dataset results with TSA fine-tuning

| Dataset       | TSA    | FES    | ConFES  | ReFES  |
|---------------|--------|--------|---------|--------|
| ilsvrc_2012   | 56.8±1.1 | 55.9±1.1 | 56.5±1.1 | 56.0±1.2 |
| omniglot      | 95.0±0.4 | 93.1±0.6 | 93.7±0.6 | 92.7±0.8 |
| aircraft      | 88.4±0.5 | 87.7±0.8 | 88.0±0.7 | 87.2±0.9 |
| cu_birds      | 81.5±0.7 | 79.3±0.9 | 79.9±0.8 | 79.4±0.9 |
| dtd           | 77.1±0.7 | 76.0±0.8 | 76.6±0.8 | 76.0±0.9 |
| quickdraw     | 82.0±0.6 | 82.6±0.6 | 83.5±0.6 | 83.3±0.6 |
| fungi         | 68.3±1.1 | 67.9±1.1 | 69.9±1.1 | 69.1±1.2 |
| vgg_flower    | 92.1±0.5 | 91.8±0.6 | 92.0±0.7 | 91.6±0.8 |
| ilsvrc_2012   | 56.8±1.1 | 55.9±1.1 | 56.5±1.1 | 56.0±1.2 |
| omniglot      | 95.0±0.4 | 93.1±0.6 | 93.7±0.6 | 92.7±0.8 |
| aircraft      | 88.4±0.5 | 87.7±0.8 | 88.0±0.7 | 87.2±0.9 |
| cu_birds      | 81.5±0.7 | 79.3±0.9 | 79.9±0.8 | 79.4±0.9 |
| dtd           | 77.1±0.7 | 76.0±0.8 | 76.6±0.8 | 76.0±0.9 |
| quickdraw     | 82.0±0.6 | 82.6±0.6 | 83.5±0.6 | 83.3±0.6 |
| fungi         | 68.3±1.1 | 67.9±1.1 | 69.9±1.1 | 69.1±1.2 |
| vgg_flower    | 92.1±0.5 | 91.8±0.6 | 92.0±0.7 | 91.6±0.8 |
| mean WG acc   | 80.2 ±  | 79.3 ±  | 80.0 ±  | 79.4 ±  |
| mean WG rank  | 2.50 ±  | 2.79 ±  | 2.26 ±  | 2.45 ±  |
| traffic_sign  | 82.8±0.9 | 84.6±1.0 | 85.2±1.0 | 85.1±1.0 |
| mscoco        | 53.8±1.1 | 53.3±1.0 | 54.6±1.0 | 54.2±1.0 |
| mnist         | 96.6±0.4 | 97.1±0.5 | 97.2±0.5 | 96.8±0.5 |
| cifar10       | 79.9±0.8 | 78.4±0.9 | 78.7±0.9 | 78.4±0.9 |
| cifar100      | 70.3±1.0 | 70.4±1.0 | 70.8±1.1 | 70.7±1.1 |
| CropDisease   | 84.4±0.8 | 88.3±0.7 | 88.2±0.7 | 88.1±0.7 |
| EuroSAT       | 89.6±0.5 | 89.1±0.6 | 89.4±0.6 | 88.9±0.7 |
| ISIC          | 48.4±0.9 | 49.1±0.9 | 49.2±1.0 | 48.5±1.0 |
| ChestX        | 27.2±0.6 | 27.9±0.6 | 27.7±0.6 | 27.7±0.6 |
| Food101       | 53.4±1.2 | 55.0±1.1 | 55.4±1.1 | 55.1±1.1 |
| mean SG acc   | 68.6 ±  | 69.3 ±  | 69.6 ±  | 69.4 ±  |
| mean SG rank  | 2.83 ±  | 2.55 ±  | 2.25 ±  | 2.36 ±  |

### Table 3: Paired t-test results with TSA fine-tuning

| WG  | TSA | FES | ConFES | ReFES |
|-----|-----|-----|--------|-------|
| TSA | -   | 1   | 2      | 2     |
| FES | 6   | -   | 8      | 2     |
| ConFES | 4 | 0   | -      | 0     |
| ReFES | 6 | 2   | -      | -     |

| SG  | TSA | FES | ConFES | ReFES |
|-----|-----|-----|--------|-------|
| TSA | -   | 6   | 8      | 6     |
| FES | 3   | -   | 6      | 3     |
| ConFES | 1 | 0   | -      | 0     |
| ReFES | 2 | 4   | 6      | -     |

**Fig. 5:** TSA weak generalisation critical difference diagram

**Fig. 6:** TSA strong generalisation critical difference diagram
Table 4: Meta-Dataset results with URL fine-tuning

| Dataset   | URL    | FES    | ConFES | ReFES  |
|-----------|--------|--------|--------|--------|
| ilsvrc2012| 56.6±1.1 | 55.4±1.1 | 56.2±1.1 | 55.6±1.2 |
| omniglot  | 94.5±0.4 | 93.0±0.6 | 94.0±0.6 | 91.2±1.0 |
| aircraft  | 87.7±0.5 | 87.2±0.8 | 87.6±0.6 | 87.1±0.9 |
| cu_birds  | 80.7±0.7 | 78.6±0.8 | 79.1±0.8 | 78.7±0.9 |
| dtd       | 76.1±0.6 | 74.3±0.8 | 74.8±0.8 | 74.5±0.8 |
| quickdraw | 82.0±0.6 | 82.3±0.6 | 83.2±0.6 | 82.8±0.6 |
| fungi     | 69.5±1.1 | 68.2±1.1 | 70.1±1.1 | 69.2±1.2 |
| vgg_flower| 91.4±0.5 | 90.6±0.6 | 90.7±0.7 | 90.3±0.7 |

Table 5: Paired $t$-test results with URL fine-tuning

| WG URL | FES | ConFES | ReFES |
|--------|-----|--------|-------|
| URL    | 6   | 7      | 2     |
| FES    | 5   | 0      | 0     |
| ConFES | 5   | 2      | -     |

| SG URL | FES | ConFES | ReFES |
|--------|-----|--------|-------|
| URL    | 4   | 3      | 1     |
| FES    | 2   | 3      | -     |
| ConFES | 1   | 3      | 7     |

Fig. 7: URL weak generalisation critical difference diagram

Fig. 8: URL strong generalisation critical difference diagram
Table 6: Meta-Dataset results with FLUTE fine-tuning

| Dataset    | FLUTE | FES  | ConFES | ReFES |
|------------|-------|------|--------|-------|
| ilsvrc_2012| 50.2 ± 1.1 | 51.7 ± 1.1 | 54.1 ± 1.1 | 53.8 ± 1.2 |
| omniglot   | 93.9 ± 0.5 | 94.4 ± 0.5 | 95.0 ± 0.5 | 94.2 ± 0.6 |
| aircraft   | 86.8 ± 0.6 | 87.2 ± 0.7 | 87.1 ± 0.8 | 87.2 ± 0.8 |
| cu_birds   | 79.3 ± 0.8 | 76.8 ± 0.9 | 78.6 ± 0.9 | 78.0 ± 0.9 |
| dtd        | 68.8 ± 0.8 | 73.0 ± 0.8 | 74.4 ± 0.9 | 74.0 ± 0.9 |
| quickdraw   | 79.1 ± 0.7 | 80.8 ± 0.6 | 82.8 ± 0.6 | 82.6 ± 0.6 |
| fungi      | 59.4 ± 1.2 | 64.6 ± 1.1 | 69.2 ± 1.1 | 68.3 ± 1.1 |
| vgg_flower | 91.0 ± 0.6 | 92.2 ± 0.6 | 92.5 ± 0.6 | 92.3 ± 0.7 |
| ilsvrc_2012| 50.2 ± 1.1 | 51.7 ± 1.1 | 54.1 ± 1.1 | 53.8 ± 1.2 |
| omniglot   | 93.9 ± 0.5 | 94.4 ± 0.5 | 95.0 ± 0.5 | 94.2 ± 0.6 |
| aircraft   | 86.8 ± 0.6 | 87.2 ± 0.7 | 87.1 ± 0.8 | 87.2 ± 0.8 |
| cu_birds   | 79.3 ± 0.8 | 76.8 ± 0.9 | 78.6 ± 0.9 | 78.0 ± 0.9 |
| dtd        | 68.8 ± 0.8 | 73.0 ± 0.8 | 74.4 ± 0.9 | 74.0 ± 0.9 |
| quickdraw   | 79.1 ± 0.7 | 80.8 ± 0.6 | 82.8 ± 0.6 | 82.6 ± 0.6 |
| fungi      | 59.4 ± 1.2 | 64.6 ± 1.1 | 69.2 ± 1.1 | 68.3 ± 1.1 |
| vgg_flower | 91.0 ± 0.6 | 92.2 ± 0.6 | 92.5 ± 0.6 | 92.3 ± 0.7 |
| mean WG acc| 76.1   | 77.6 | 79.2   | 78.8 |
| mean WG rank| 3.13  | 2.75 | 2.00  | 2.13 |

Table 7: Paired t-test results with FLUTE fine-tuning

| WG   | FLUTE | FES  | ConFES | ReFES |
|------|-------|------|--------|-------|
| FLUTE| -     | 7    | 6      | 6     |
| FES  | 7     | -    | 5      | -     |
| ConFES| 0    | 0    | -      | 0     |
| ReFES| 0     | 7    | -      | -     |

Fig. 9: FLUTE weak generalisation critical difference diagram

Fig. 10: FLUTE strong generalisation critical difference diagram
5.1 Meta-Dataset results

Results are organised by fine-tuning algorithms used, to provide a comparison between different meta-learning algorithms with the same fine-tuning scheme. The universal model of URL (W. Li et al., 2021), applied with TSA fine-tuning (W. Li et al., 2022), is the most recent and strongest CDFSML approach in the literature. Hence, we compare to this universal-model approach first, applying TSA fine-tuning in our FES methods as well in this comparison. Following that, we present experiments with the simpler fine-tuning approach used in the original URL (W. Li et al., 2021) paper. Finally, we evaluate FLUTE (Triantafillou et al., 2021) fine-tuning, which fine-tunes batch norm parameters only, and compare to the FLUTE universal template model.

Results with TSA fine-tuning are shown in Table 2, and paired t-test results based on the 600 individual accuracy values per dataset are shown in Table 3. Results with URL fine-tuning are shown in Tables 4 and 5, and those with FLUTE fine-tuning are shown in Tables 6 and 7.

In these tables, mean accuracy over 600 episodes and 95% confidence intervals are shown for each algorithm and dataset, and weak and strong generalisation accuracy and ranks averaged over all individual episodes are listed below the datasets. The best result of each row is shown in bold. If a paired t-test between a FES algorithm and the corresponding universal model/template (in the leftmost column) returns a p value less than 0.05, the null hypothesis (that there is no statistically significant difference) is rejected, and the FES result is marked with either ◦ if it has higher accuracy, or • if its competitor has higher accuracy.

The tables showing paired t-test results are split by weak generalisation (the eight source domains) and strong generalisation (the ten target domains). Each value indicates the number of datasets where the algorithm in the value’s column significantly outperforms the algorithm in its row according to the paired t-test.

Figures 5, 6, 7, 8, 9, and 10 are critical difference diagrams produced by the Wilcoxon test applied with α = 0.05 and the Holm correction, where algorithms are ranked using all relevant accuracy values (8 datasets × 600 episodes for weak generalisation, and 10 datasets × 600 episodes for strong generalisation). Algorithms with no statistically significant difference are grouped via horizontal lines.

When using the same fine-tuning scheme, FES and its variants outperform their competitor meta-learning algorithms—building a universal model using knowledge distillation for URL and TSA, and training a universal template with FiLM layers for FLUTE—in strong generalisation, where learning problems qualify as being cross-domain. The FES algorithms achieve better average accuracy and obtain more wins than losses in paired t-tests. They also rank higher than their competitors in the critical difference diagram.

Considering results with all three fine-tuning methods, the FES algorithms consistently outperform their competitors by a substantial margin on traffic_sign, CropDisease, and Food101, while being outperformed on cifar10 and
cifar100. This phenomenon may indicate that FES and its variants perform better in domains that are more specialised, while their competitors gain an edge on datasets more similar to ImageNet, such as the CIFAR datasets. This speculation is supported by the fact that the competitor methods artificially attach greater importance to ImageNet when their universal models are obtained (W. Li et al., 2021, 2022; Triantafillou et al., 2021).

Considering all results of the three FES variants, ConFES is most favourable. ConFES exhibits the best strong generalisation accuracy and maintains a smaller number of meta-classifier parameters than FES and ReFES.
Table 8: Percentage of snapshots omitted by the meta-classifier

| Dataset         | FES | ConFES | ReFES |
|-----------------|-----|--------|-------|
|                 | TSA URL FLUTE | TSA URL FLUTE | TSA URL FLUTE |
| ilsvrc 2012     | 15.9 | 16.1 | 2.7 | 27.7 | 29.1 | 26.6 | 43.6 | 45.2 | 44.8 |
| omniglot        | 4.9  | 4.5  | 0.8 | 26.9 | 30.3 | 31.5 | 37.2 | 39.8 | 40.7 |
| aircraft        | 20.7 | 19.5 | 2.9 | 30.5 | 31.6 | 31.8 | 42.8 | 44.6 | 38.3 |
| cu_birds        | 23.8 | 23.6 | 4.4 | 35.9 | 37.5 | 38.6 | 47.2 | 46.6 | 44.0 |
| dtd             | 14.1 | 12.6 | 2.7 | 26.2 | 30.5 | 23.0 | 35.8 | 42.7 | 38.2 |
| quickdraw       | 7.9  | 6.7  | 0.3 | 23.9 | 24.4 | 25.5 | 36.5 | 39.4 | 47.5 |
| fungi           | 21.2 | 23.3 | 2.6 | 38.2 | 37.0 | 35.6 | 47.6 | 47.7 | 46.0 |
| vgg_flower      | 7.8  | 5.9  | 0.9 | 20.3 | 20.1 | 24.6 | 29.4 | 30.8 | 33.1 |
| traffic_sign    | 17.5 | 18.2 | 1.9 | 29.5 | 35.7 | 29.8 | 34.0 | 57.8 | 72.0 |
| mscoco          | 10.2 | 6.9  | 0.7 | 22.5 | 18.6 | 20.5 | 39.8 | 38.7 | 44.1 |
| mnist           | 3.6  | 3.1  | 0.7 | 19.2 | 23.3 | 30.3 | 26.3 | 30.8 | 38.9 |
| cifar10         | 16.6 | 11.9 | 2.3 | 27.7 | 24.2 | 29.1 | 41.4 | 41.3 | 43.8 |
| cifar100        | 8.8  | 8.2  | 1.0 | 20.2 | 18.2 | 19.8 | 39.4 | 41.0 | 46.2 |
| CropDisease     | 1.5  | 1.1  | 0.1 | 12.6 | 11.3 | 13.8 | 30.1 | 28.3 | 36.2 |
| EuroSAT         | 10.6 | 6.7  | 0.4 | 27.4 | 22.4 | 19.0 | 34.6 | 34.3 | 37.7 |
| ISIC            | 14.5 | 11.3 | 0.9 | 24.4 | 29.6 | 18.8 | 38.2 | 42.4 | 37.8 |
| ChestX          | 29.6 | 18.2 | 5.3 | 38.9 | 29.9 | 23.0 | 48.8 | 47.9 | 46.9 |
| Food101         | 12.9 | 7.5  | 1.7 | 21.6 | 17.9 | 21.1 | 52.2 | 40.3 | 38.9 |

5.2 Weight visualisation

Weights of the FES, ConFES, and ReFES kernels after fine-tuning with TSA on traffic_sign are visualised in Figures 11, 12, and 13, which are averaged over 600 episodes.

ConFES maintains two kernels: a low-level depthwise 1D convolutional kernel (12a) and a high-level global kernel (12b). The two kernels can be expanded back into a global kernel (12c) for interpretation because the output of the convolutional kernel 12a serves as direct input to the global kernel 12b, without any intermediate non-linear activation. Figure 12 demonstrates how ConFES emulates a 328-parameter FES kernel with only 144 parameters. The stepped pattern in the expanded ConFES kernel, where every fourth snapshot is assigned relatively greater weight than its neighbours, is an artefact of 1D convolution—with a kernel size of 9 and a stride size of 4, this pattern results from kernel overlaps.

FES determines that the fine-tuned ilsvrc_2012 (ImageNet) and quickdraw backbones are the most prominent contributors to its predictions, indicated by the dark regions on the right end of these two backbones’ rows in Figure 11. ConFES and ReFES arrive at similar conclusions regarding contributors, but exhibit sparser selections of snapshots, as Figures 12 and 13 can be seen as sparsified versions of Figure 11.

Additional heatmaps visualising kernel weights on the other target domains are in Appendix A, shown by Figures 14-40.
Table 9: Ablation results with TSA fine-tuning

| Dataset         | first no CV | last ConFES | first CV | last ConFES |
|-----------------|-------------|-------------|----------|-------------|
| ilsvrc_2012     | 51.3±1.2    | 48.9±1.2    | 55.2±1.2 | 52.6±1.1    | 51.3±1.1    | 56.5±1.1    |
| omniglot        | 94.7±0.4    | 90.0±0.7    | 90.9±0.7 | 94.7±0.4    | 91.2±0.7    | 93.7±0.6    |
| aircraft        | 86.4±0.7    | 79.6±1.2    | 82.2±0.9 | 87.1±0.6    | 85.9±0.8    | 88.0±0.7    |
| cu_birds        | 74.4±0.9    | 70.8±1.1    | 77.3±1.0 | 76.2±0.8    | 76.7±0.9    | 79.9±0.8    |
| dtb             | 69.0±0.8    | 73.9±0.9    | 76.6±0.8 | 69.6±0.7    | 74.4±0.8    | 76.6±0.8    |
| quickdraw       | 81.8±0.6    | 76.0±0.8    | 83.5±0.6 | 82.0±0.6    | 77.4±0.7    | 83.5±0.6    |
| fungi           | 64.5±1.2    | 53.9±1.2    | 64.0±1.3 | 65.8±1.1    | 60.0±1.1    | 69.9±1.1    |
| vgg_flower      | 88.1±0.6    | 89.7±0.8    | 89.6±0.9 | 88.2±0.6    | 91.0±0.7    | 92.0±0.7    |
| mean WG acc     | 76.3        | 72.9        | 77.4     | 77.0        | 76.0        | 80.0        |
| traffic_sign    | 45.0±1.0    | 84.0±1.0    | 84.8±0.9 | 45.3±1.0    | 84.3±1.0    | 85.2±1.0    |
| mnist           | 43.7±1.0    | 47.1±1.0    | 54.6±1.0 | 44.2±1.0    | 48.9±1.0    | 54.6±1.0    |
| cifar10         | 95.6±0.4    | 96.5±0.5    | 96.2±0.6 | 95.6±0.4    | 96.7±0.5    | 97.2±0.5    |
| cifar100        | 62.8±0.8    | 71.0±0.9    | 78.3±0.9 | 63.5±0.8    | 74.0±0.9    | 78.7±0.9    |
| CropDisease     | 52.4±1.1    | 63.1±1.1    | 70.4±1.1 | 53.3±1.1    | 66.0±1.1    | 70.8±1.1    |
| EuroSAT         | 81.7±0.8    | 86.3±0.7    | 87.1±0.7 | 81.9±0.7    | 86.7±0.7    | 88.2±0.7    |
| ISIC            | 76.8±0.6    | 87.0±0.7    | 89.1±0.6 | 76.8±0.6    | 87.4±0.6    | 89.4±0.6    |
| ChestX          | 45.6±0.8    | 46.8±0.9    | 48.2±0.9 | 45.5±0.8    | 47.0±0.9    | 49.2±1.0    |
| Food101         | 24.6±0.5    | 27.3±0.6    | 27.2±0.6 | 24.4±0.5    | 27.3±0.6    | 27.7±0.6    |
| mean SG acc     | 49.3±1.1    | 50.0±1.2    | 54.3±1.2 | 50.7±1.1    | 52.5±1.1    | 55.4±1.1    |

5.3 Snapshot omission

As FES kernels are constrained to be non-negative by clipping their weights with ReLU, some snapshots may have their corresponding weights set to 0 after clipping, which means logits from these snapshots do not contribute to the aggregated meta-logits, and these snapshots can be omitted, i.e., they do not need to be saved and are not used for inference.

Table 8 shows the average percentage of snapshots omitted by a FES, ConFES, or ReFES meta-classifier, using TSA, URL, or FLUTE fine-tuning. Note that ConFES omission rates are computed using the expanded kernel, because 0 values need to exist in the expanded kernel, instead of merely in one of the ConFES hierarchical kernels, for the corresponding snapshots to be omitted. A higher omission percentage is considered better because omitting snapshots saves storage space and inference computation. Among the three methods, ReFES achieves the highest percentage of omission, generally between 30% and 50%, followed by ConFES, which achieves 20% to 35% omission in general, while FES achieves the least amount of omission, mostly below 25%. The high ReFES omission percentage can be attributed to fused lasso regularisation used in ReFES, which encourages less relevant weights to reduce to 0. Due to their higher snapshot omission rates, ConFES and ReFES are preferable to FES.
6 Ablation study

We perform an ablation study with ConFES, by removing cross-validation from the framework, and/or using only the first or last snapshots in fine-tuning. When cross-validation is not used, training logits for the meta-classifier are extracted from the support set using snapshots fine-tuned on the entire support set, akin to how one-shot episodes are handled in Section 3.5. When using only the first or last snapshots, the meta-classifier is a degenerate weight kernel with a singleton dimension for fine-tuning iterations, simply containing one weight value for each backbone. Results are shown in Tables 9, 10, and 11, organised by the fine-tuning algorithm used.

The results show that methods using cross-validation outperform their counterparts without cross-validation. Moreover, applying ConFES to all snapshots saved throughout the fine-tuning process achieves better performance than only using snapshots at the beginning or end of fine-tuning. This trend is consistent in terms of mean strong generalisation performance across all three fine-tuning algorithms. It is worth noting that cross-validation is helpful even when only using the first snapshots before any fine-tuning, because the training logits are computed using a nearest centroid classifier, and cross-validation keeps the support instances for logit extraction separate from those used to compute the centroids, hence avoiding instance re-use and reducing overfitting. Applying ConFES to all snapshots outperforms only using the last snapshots at the end of fine-tuning, indicating that snapshots during fine-tuning can positively contribute to the meta-classifier’s predictions.
Table 11: Ablation results with FLUTE fine-tuning

| Dataset         | first ± | no CV ± | ConFES ± | first ± | CV ± | ConFES ± |
|-----------------|---------|---------|----------|---------|-----|----------|
| ilsvrc_2012     | 47.3±1.1| 48.0±1.1| 52.7±1.2| 49.2±1.1| 49.8±1.1| 54.1±1.1|
| omniglot        | 93.3±0.6| 93.1±0.6| **95.2±0.4**| 94.1±0.5| 94.1±0.6| 95.0±0.5|
| aircraft        | 84.7±0.9| 82.7±1.0| 86.3±1.0| 86.5±0.7| 86.4±0.8| **87.1±0.8**|
| cu_birds        | 69.7±1.1| 70.7±1.1| 77.2±0.9| 73.1±0.9| 75.1±0.9| **78.6±0.9**|
| dtid            | 68.6±0.8| 71.4±0.8| **74.6±0.8**| 69.0±0.8| 72.0±0.8| 74.4±0.9|
| quickdraw       | 78.5±0.7| 77.7±0.7| 82.6±0.6| 79.4±0.7| 78.4±0.7| **82.8±0.6**|
| fungi           | 58.7±1.3| 55.8±1.2| 66.9±1.2| 61.8±1.2| 60.0±1.1| **69.2±1.1**|
| vgg_flower      | 90.3±0.7| 91.3±0.7| 92.1±0.6| 90.8±0.6| 92.0±0.6| **92.5±0.6**|
| mean WG acc     | 73.9    | 73.8    | 78.5     | 75.5    | 76.0 | 79.2     |
| traffic_sign    | 50.3±1.1| 69.4±1.1| 68.4±1.2| 50.7±1.1| 69.8±1.1| **70.9±1.1**|
| mnist           | 41.0±1.1| 45.6±1.1| 50.6±1.1| 41.7±1.1| 46.4±1.1| **51.9±1.1**|
| cifar10         | 95.7±0.4| 96.8±0.5| **97.6±0.3**| 95.8±0.4| 97.1±0.4| **97.6±0.4**|
| cifar100        | 64.9±0.9| 70.5±0.9| **75.4±0.8**| 65.8±0.9| 71.4±0.9| 75.3±0.9|
| CropDisease     | 55.6±1.1| 61.5±1.1| 66.0±1.1| 57.0±1.1| 63.1±1.1| **66.9±1.0**|
| EuroSAT         | 80.9±0.8| 85.2±0.7| 86.0±0.7| 81.2±0.8| 85.5±0.7| **86.2±0.7**|
| ISIC            | 81.7±0.6| 86.8±0.6| 87.8±0.6| 81.6±0.6| 86.8±0.6| **88.1±0.6**|
| ChestX          | 47.4±0.9| 46.4±0.9| 44.7±0.9| 47.4±0.9| 46.8±0.9| **48.8±1.0**|
| Food101         | 25.8±0.5| **27.8±0.6**| 27.2±0.6| 25.8±0.5| **27.8±0.6**| 27.5±0.6|
| mean SG acc     | 58.7    | 63.7    | 65.4     | 59.3    | 64.4 | **66.5** |

7 Heterogeneous backbones

FES and its variants operate in logit space, which means they are myopic to the architecture and feature size of each backbone. Therefore, they can naturally work with heterogeneous backbone collections. We demonstrate this by replacing the ResNet18 ImageNet backbone in the source domain collection with a more advanced Small EfficientNetV2 model (Tan & Le, 2021) pretrained on the 21K-class version of ImageNet, while keeping the seven other source domain ResNet18 backbones unchanged. The Small EfficientNetV2 model produces feature vectors of length 1280, as opposed to feature vectors of length 512 generated by ResNet18.

URL-style fine-tuning is used, i.e., a square matrix is used for feature projection and the matrix is initialised as an identity matrix. The results are shown in Table 12, and are compared to results of all eight backbones being ResNet18 models. Usage of the EfficientNetV2 model consistently improves FES performance in both weak and strong generalisation. Note that the evaluation’s main purpose is to show FES compatibility with heterogeneous model zoos, and its results are not directly comparable to the main results, because the 21K-class ImageNet dataset used to pretrain the EfficientNetV2 model contains the Meta-Dataset ImageNet test split, which makes the ImageNet evaluation over-optimistic, and test classes in the other domains may also be present in the 21K pretraining classes.

Since the EfficientNetV2 model is much more advanced than ResNet18, we investigate whether it dominates the backbone collection and effectively makes
Table 12: Results of replacing the ResNet18 ImageNet backbone with a Small EfficientNetV2 pretrained on the 21K version of ImageNet, while the other seven backbones remain the same.

| Dataset       | FES  | ConFES | ReFES | FES  | ConFES | ReFES |
|---------------|------|--------|-------|------|--------|-------|
| ilsvrc 2012   | 55.4±1.1 | 56.2±1.1 | 55.6±1.2 | 61.6±1.0 | 63.7±1.0 | 63.4±1.1 |
| omniglot      | 93.0±0.6 | **94.0±0.6** | 91.2±1.0 | 92.8±0.6 | **94.0±0.6** | 91.0±1.1 |
| aircraft      | 87.2±0.8 | 87.6±0.6 | 87.1±0.9 | 87.4±0.7 | **87.9±0.6** | 87.1±0.9 |
| cu_birds      | 78.6±0.8 | 79.1±0.8 | 78.7±0.9 | 79.0±0.8 | **80.9±0.8** | 80.0±0.9 |
| dtd           | 74.3±0.8 | 74.8±0.8 | 74.5±0.8 | 79.6±0.8 | **81.5±0.7** | 81.0±0.8 |
| quickdraw     | 82.3±0.6 | 83.2±0.6 | 82.8±0.6 | 82.5±0.6 | **83.3±0.6** | 83.0±0.6 |
| fungi         | 68.2±1.1 | 70.1±1.1 | 69.2±1.2 | 68.7±1.1 | **70.4±1.1** | 69.8±1.2 |
| vgg_flower    | 90.6±0.6 | 90.7±0.7 | 90.3±0.7 | 95.0±0.5 | **97.0±0.3** | 95.6±0.7 |
| mean WG acc   | 78.7 | 79.5 | 78.7 | 80.8 | **82.3** | 81.3 |
| traffic_sign  | 63.8±1.2 | 64.8±1.2 | 64.8±1.2 | 64.8±1.1 | **65.4±1.1** | 65.2±1.2 |
| mscoco        | 51.9±1.0 | 52.9±1.0 | 52.6±1.0 | 57.9±1.0 | **60.2±1.0** | 60.0±1.0 |
| mnist         | 96.5±0.5 | 96.5±0.5 | 96.2±0.6 | **96.6±0.5** | 96.6±0.5 | 96.3±0.6 |
| cifar10       | 71.6±0.8 | 72.1±0.8 | 71.7±0.9 | 81.9±0.8 | **83.0±0.8** | 82.6±0.8 |
| cifar100      | 62.4±1.1 | 63.0±1.1 | 62.7±1.1 | 72.7±1.0 | **74.1±1.0** | 73.8±1.0 |
| CropDisease   | 87.2±0.7 | 87.2±0.7 | 87.0±0.7 | 89.3±0.6 | **89.5±0.6** | 89.3±0.6 |
| EuroSAT       | 86.0±0.6 | 86.2±0.6 | 85.9±0.6 | 88.6±0.6 | **89.1±0.6** | 88.6±0.7 |
| ISIC          | 48.6±0.9 | 48.6±0.9 | 48.5±0.9 | **49.0±0.9** | 48.7±0.9 | 48.5±0.9 |
| ChestX        | 27.0±0.6 | 26.9±0.6 | 26.9±0.6 | **27.0±0.6** | 26.7±0.6 | 26.8±0.5 |
| Food101       | 54.0±1.1 | 54.1±1.1 | 53.8±1.1 | 60.8±1.0 | **61.8±1.0** | 61.5±1.1 |
| mean SG acc   | 64.9 | 65.2 | 65.0 | 68.9 | **69.5** | 69.3 |

the other ResNet18 backbones irrelevant, by performing FES using the single EfficientNetV2 backbone, with results in Table 13. Interestingly, all three FES variants obtain very similar accuracy when applied to only one EfficientNetV2 backbone, while their differences are shown more clearly when applied to a collection of eight backbones. Although using only EfficientNetV2 leads to better performance in a number of ImageNet-adjacent domains, e.g., ilsvrc 2012, dtd, vgg_flower, mscoco, and cifar10, it under-performs in most other domains, especially those significantly different from ImageNet, e.g., omniglot, aircraft, quickdraw, fungi, traffic_sign, mnist, CropDisease, ISIC, and ChestX.

Our EfficientNetV2 evaluation indicates: 1) FES and its variants are compatible with heterogeneous backbone collections, and 2) they are robust to discrepancies in backbone architectures and able to select relevant models from a diverse model zoo.

8 Limitations and discussion

FES requires no universal backbone, which means the meta-training phase only requires obtaining a collection of backbones. The cost for this is reduced to zero if pretrained backbones are readily available. However, FES is more expensive in the meta-testing phase in terms of both computation and storage, as it needs to fine-tune each backbone and save their snapshots, instead of utilising a single universal backbone. The good performance of FES could be attributed
to its increased capacity, as it maintains individual backbones instead of a single universal backbone. In the context of Meta-Dataset, FES maintains eight backbones, which means 8× parameters compared to a universal model of the same architecture. We investigate larger universal models with capacities comparable to FES.

Originally, W. Li et al. (2021) distils eight ResNet18 backbone into a universal ResNet18 backbone. We distilled a universal ResNet152 (He et al., 2016) backbone using the same process. ResNet18 has 11M parameters while ResNet152 has 60M. We elected to use the same eight ResNet18 backbones for distillation, because pretraining eight ResNet152 backbones from scratch is prohibitively expensive for us, and this avoids introducing a confounding factor to meta-model evaluation because different base-model architectures may encompass source domain semantics differently. We also pretrained a universal ResNet152 model using “vanilla” multi-domain learning (MDL), i.e., one feature extractor is pretrained with all eight source domains’ data using eight classification heads, one for each domain. Compared to official ResNet18 URL training, we halved the mini-batch size (and doubled the number of iterations) to fit ResNet152 URL or MDL training in the 4GB memory of an NVIDIA A6000 GPU—the most advanced at our disposal. Tables 14 and 15 show their results with URL or TSA fine-tuning respectively, and compare them to using the official ResNet18 URL model, as well as FES variants with ResNet18 backbone collections. As TSA fine-tuning has high memory consumption, we

Table 13: Comparison between applying FES to an ImageNet-pretrained EfficientNetV2 backbone alone and applying FES to a backbone collection containing it and the seven other ResNet18 source domain backbones.

| Dataset       | EfficientNetV2 only | EfficientNetV2 and ResNet18s |
|---------------|---------------------|-----------------------------|
|               | FES | ConFES | ReFES | FES | ConFES | ReFES |
| ilsvrc2012    | 63.9±1.0 | 63.9±1.0 | 63.9±1.0 | 61.6±1.0 | 63.7±1.0 | 63.1±1.1 |
| omniglot      | 57.8±1.3 | 57.9±1.3 | 57.2±1.3 | 92.8±0.6 | 94.0±0.6 | 91.0±1.1 |
| aircraft      | 63.6±1.0 | 63.7±1.0 | 63.5±1.0 | 87.4±0.7 | 87.9±0.6 | 87.1±0.9 |
| cu_birds      | 75.9±0.8 | 76.0±0.8 | 75.9±0.8 | 79.0±0.8 | 80.9±0.8 | 80.0±0.9 |
| dtd           | 82.1±0.6 | 82.1±0.7 | 82.1±0.6 | 79.6±0.8 | 81.5±0.7 | 81.0±0.8 |
| quickdraw     | 59.8±1.0 | 60.0±1.0 | 59.7±1.0 | 82.5±0.6 | 83.3±0.6 | 83.0±0.6 |
| fungi         | 50.9±1.2 | 51.0±1.2 | 50.8±1.2 | 68.7±1.1 | 70.4±1.1 | 69.8±1.2 |
| vgg_flower    | 97.2±0.2 | 97.2±0.2 | 97.2±0.2 | 95.0±0.5 | 97.0±0.3 | 95.6±0.7 |
| mean WG acc   | 68.9 | 69.0 | 68.8 | 80.8 | 82.3 | 81.3 |
| traffic_sign  | 60.3±1.2 | 60.2±1.2 | 60.4±1.2 | 64.8±1.1 | 65.4±1.1 | 65.2±1.2 |
| mscoco        | 60.9±0.9 | 61.2±0.9 | 60.6±0.9 | 57.9±1.0 | 60.2±1.0 | 60.0±1.0 |
| mnist         | 87.1±0.7 | 87.1±0.7 | 87.0±0.7 | 96.6±0.5 | 96.6±0.5 | 96.3±0.6 |
| cifar10       | 83.3±0.6 | 83.3±0.6 | 83.3±0.6 | 81.9±0.8 | 83.0±0.8 | 82.6±0.8 |
| cifar100      | 73.8±0.9 | 73.7±0.9 | 73.7±0.9 | 72.7±1.0 | 74.1±1.0 | 73.8±1.0 |
| CropDisease   | 85.4±0.7 | 85.4±0.7 | 85.3±0.7 | 89.3±0.6 | 89.5±0.6 | 89.3±0.6 |
| EuroSAT       | 87.0±0.6 | 87.0±0.6 | 86.8±0.6 | 88.6±0.6 | 89.1±0.6 | 88.6±0.7 |
| ISIC          | 46.2±0.9 | 46.0±0.9 | 46.4±0.9 | 49.0±0.9 | 48.7±0.9 | 48.5±0.9 |
| ChestX        | 25.2±0.5 | 25.2±0.5 | 25.1±0.6 | 27.0±0.6 | 26.7±0.6 | 26.8±0.5 |
| Food101       | 61.1±1.0 | 61.1±1.0 | 61.0±1.0 | 60.8±1.0 | 61.8±1.0 | 61.5±1.1 |
| mean SG acc   | 67.0 | 67.0 | 67.0 | 68.9 | 69.5 | 69.3 |
Table 14: ResNet152 feature extractors with URL fine-tuning.

| Dataset       | URL.18 | MDL.152 | URL.152 | FES.18 | ConFES.18 | ReFES.18 |
|---------------|--------|---------|---------|--------|-----------|----------|
| islvrc_2012   | 56.6±1 | 59.3±1  | 59.3±1  | 55.4±1 | 56.2±1.1  | 55.6±1.2 |
| omniglot      | 94.5±0.4| 94.5±0.4| 94.8±0.4| 93.0±0.6| 94.0±0.6  | 91.2±1.0 |
| aircraft      | 87.7±0.5| 90.7±0.4| 91.3±0.4| 87.2±0.8| 87.6±0.6  | 87.1±0.9 |
| cu_birds      | 80.7±0.7| 85.0±0.6| 84.9±0.6| 78.6±0.8| 79.1±0.8  | 78.7±0.9 |
| dtd           | 76.1±0.6| 77.7±0.6| 78.3±0.6| 74.3±0.8| 74.8±0.8  | 74.5±0.8 |
| quickdraw     | 82.0±0.6| 83.3±0.6| 83.3±0.6| 82.3±0.6| 83.2±0.6  | 82.8±0.6 |
| fungi         | 69.5±1.1| 73.3±1.0| 74.9±1.0| 68.2±1.1| 70.1±1.1  | 69.2±1.2 |
| vgg_flower    | 91.4±0.5| 93.3±0.4| 91.9±0.5| 90.6±0.6| 90.7±0.7  | 90.3±0.7 |
| mean WG acc   | 79.8   | 82.1    | 82.3    | 78.7   | 79.5      | 78.7     |
| mean SG acc   | 63.5   | 63.6    | 64.9    | 64.9   | 65.2      | 65.0     |

Table 14 compares the cost of FES inference using an NVIDIA A6000 GPU to that of URL ResNet18 and ResNet152 backbones. TSA fine-tuning is used by all methods in this table. It is worth pointing out that due to the few-shot nature of each episode, meta-testing is generally not time consuming. Table 16 represents the approximate upper bound of FES computation cost, because 1) the time presented in the table was measured using the largest traffic task episode in our cached sample, which contains 497 support instances, whereas smaller episodes consume less time, 2) URL and FLUTE fine-tuning are much less time-consuming than TSA, and 3) Section 5.3 shows that a portion of the snapshots does not in fact need to be computed and stored. FES requires approximately $2 \times K$ as much backpropagation as a universal backbone fine-tuned once, where $2$ represents one fine-tuning run on the cross-validated support set (performed in two splits) and another on the full
support set, and $K$ represents the number of backbones. This is reflected in Table 16 as fine-tuning time for the FES methods is approximately 16 times that of fine-tuning the URL ResNet18 model. Time required to train a FES or ConFES meta-classifier is relatively trivial, while ReFES requires more time to determine its regularisation strength using grid search with cross-validation. FES stores multiple snapshots of each backbone during fine-tuning, but not all model parameters need to be saved. Only weights that are updated during fine-tuning need to be saved in snapshots, as the other unchanged weights can be loaded from the original backbone. Common CDFSML fine-tuning algorithms only update a relatively small set of weights: FLUTE fine-tunes batch normalisation weights, URL fine-tunes a feature projection, and TSA fine-tunes channel projections and a feature projection. Therefore, FES snapshots are normally lightweight. Table 16 shows that FES with TSA fine-tuning needs to store approximately 580M parameters—2.32GB—which can fit in most modern GPUs during inference. As FES can fine-tune its backbones sequentially, its memory requirement is comparable to fine-tuning a single backbone with the same method. On the other hand, FES can easily be parallelised to fine-tune multiple backbones at once, should multiple GPUs be available. Even though a universal backbone only needs to be trained once, this training process may take days (for ResNet18) to weeks (for ResNet152) on an NVIDIA

Table 15: ResNet152 feature extractors with TSA fine-tuning. Note that the “URL” in the column titles refers to the universal representation learning process used to meta-train the models, not the feature projection fine-tuning used by the URL algorithm during meta-test. All methods in this table use TSA fine-tuning.

| Dataset         | URL_18   | MDL_152 | URL_152 | FES_18   | ConFES_18 | ReFES_18 |
|-----------------|----------|---------|---------|----------|-----------|----------|
| ilsvrc_2012     | 56.8±1.1 | 59.3±1.1| 59.9±1.1| 55.9±1.1 | 56.5±1.1  | 56.0±1.2 |
| omniglot        | 95.0±0.4 | 94.7±0.4| 95.0±0.4| 93.1±0.6 | 93.7±0.6  | 92.7±0.8 |
| aircraft        | 88.4±0.5 | 91.2±0.4| 92.2±0.4| 87.7±0.8 | 88.0±0.7  | 87.2±0.9 |
| cu_birds        | 81.5±0.7 | 84.8±0.6| 85.0±0.6| 79.3±0.9 | 79.9±0.8  | 79.4±0.9 |
| dtd             | 77.1±0.7 | 78.9±0.7| 79.2±0.7| 76.0±0.8 | 76.6±0.8  | 76.0±0.9 |
| quickdraw        | 82.0±0.6 | 83.4±0.6| 83.4±0.6| 82.6±0.6 | 83.5±0.6  | 83.3±0.6 |
| fungi           | 68.3±1.1 | 73.0±1.0| 74.5±1.0| 67.9±1.1 | 69.9±1.1  | 69.1±1.2 |
| vgg_flower      | 92.1±0.5 | 93.4±0.5| 92.2±0.6| 91.8±0.6 | 92.0±0.7  | 91.6±0.8 |
| mean WG acc     | 80.2     | 82.3    | 82.7    | 79.3     | 80.0      | 79.4     |
| traffic_sign    | 82.8±0.9 | 76.4±1.0| 80.0±0.9| 84.6±1.0 | 85.2±1.0  | 85.1±1.0 |
| mscoco          | 53.8±1.1 | 54.1±1.0| 56.9±1.0| 53.3±1.0 | 54.6±1.0  | 54.2±1.0 |
| mnist           | 96.6±0.4 | 95.5±0.5| 95.4±0.5| 97.1±0.5 | 97.2±0.5  | 96.8±0.5 |
| cifar10         | 79.9±0.8 | 79.3±0.8| 81.6±0.7| 78.4±0.9 | 78.7±0.9  | 78.4±0.9 |
| cifar100        | 70.3±1.0 | 70.5±1.0| 73.1±1.0| 70.4±1.0 | 70.8±1.1  | 70.7±1.1 |
| CropDisease     | 84.4±0.8 | 82.7±0.8| 84.4±0.7| 88.3±0.7| 88.2±0.7  | 88.1±0.7 |
| EuroSAT         | 89.6±0.5 | 88.6±0.6| 89.6±0.5| 89.1±0.6 | 89.4±0.6  | 88.9±0.7 |
| ISIC            | 48.4±0.9 | 45.6±0.9| 47.3±0.9| 49.1±0.9 | 49.2±1.0  | 48.5±1.0 |
| ChestX          | 27.2±0.6 | 25.5±0.6| 27.3±0.6| 27.9±0.6| 27.7±0.6  | 27.7±0.6 |
| Food101         | 53.4±1.2 | 56.2±1.1| 57.3±1.1| 55.0±1.1| 55.4±1.1  | 55.1±1.1 |
| mean SG acc     | 68.6     | 67.4    | 69.3    | 69.3     | 69.6      | 69.4     |
Table 16: Computational resource consumption of FES variants using TSA fine-tuning, compared to the official TSA algorithm. From left to right: CDFSML method, fine-tuning time, meta-classifier training time, number of pretrained parameters that are frozen during fine-tuning, number of trainable parameters during fine-tuning, number of parameters that need to be stored, number of meta-classifier parameters, and amount of GPU memory required for fine-tuning.

| method     | FT time | MC time | frozen P | trainable P | stored P | MC P | FT memory |
|------------|---------|---------|----------|-------------|----------|------|-----------|
| URL_18     | 10.13s  | -       | 11M      | 1.5M        | 11M + 1.5M | -    | 8.2GB     |
| URL_152    | 103.88s | -       | 60M      | 7.3M        | 60M + 7.3M | -    | 32.7GB    |
| FES_18     | 0.06s   | 11M×8   | 1.5M×8   | 11M×8 +     | 328      | 8.3GB |
| ConFES_18  | 0.06s   | 11M×8   | 1.5M×8   | 1.5M×8×41   | 328      | 8.3GB |
| ReFES_18   | 9.45s   |         |          |             |          |      |           |

Table 17: Comparing the official URL model to a URL model distilled without favouring ImageNet.

| Dataset      | URL_official | URL_equal |
|--------------|--------------|-----------|
| ilsvrc_2012  | 56.6±1.1     | 52.6±1.1  |
| omniglot     | 94.5±0.4     | 95.0±0.4  |
| aircraft     | 87.7±0.5     | 88.6±0.5  |
| cu_birds     | 80.7±0.7     | 80.0±0.7  |
| dtd          | 76.1±0.6     | 72.0±0.7  |
| quickdraw    | 82.0±0.6     | 82.1±0.6  |
| fungi        | 69.5±1.1     | 68.4±1.1  |
| vgg_flower   | 91.4±0.5     | 89.5±0.6  |
| mean WG acc  | 79.8         | 78.5      |
| traffic_sign | 62.6±1.2     | 63.0±1.2  |
| mscoco       | 52.7±1.0     | 47.2±1.0  |
| mnist        | 94.6±0.4     | 95.1±0.4  |
| cifar10      | 71.4±0.8     | 66.5±0.8  |
| cifar100     | 62.6±1.1     | 56.9±1.1  |
| CropDisease  | 80.5±0.8     | 79.9±0.8  |
| EuroSAT      | 86.6±0.5     | 83.5±0.6  |
| ISIC         | 45.5±0.8     | 44.8±0.8  |
| ChestX       | 26.6±0.6     | 26.6±0.6  |
| Food101      | 51.9±1.1     | 49.0±1.1  |
| mean SG acc  | 63.5         | 61.3      |

A6000 GPU; if an individual backbone is added or updated, training of a universal backbone needs to be performed again.

The official URL model was distilled in a process favouring ImageNet by including as many ImageNet instances as the other seven source domains combined in each mini-batch (W. Li et al., 2021), and we distilled an alternative URL model while treating all source domains equally. Their comparison is shown in Table 17. The official model performs better in a majority of domains. This indicates that URL distillation may require external knowledge to focus
on the right domains to achieve optimal performance. FES and its variants treat all backbones equally \textit{a priori} and determine their task-specific relevance based purely on the support set.

9 Conclusion

We present the stacking-based CDFSML method FES and the variants ConFES and ReFES. The FES algorithms create snapshots from fine-tuning independent backbones on the support set, use cross-validation to avoid overfitting from support data reuse, and train a simple meta-classifier to appropriately weight the snapshots. FES, ConFES, and ReFES advance the state-of-the-art on the Meta-Dataset benchmark.

Perhaps more importantly, the FES approaches have some practical advantages in real-world scenarios compared to recent methods based on universal models. FES can work with out-of-the-box heterogeneous backbones and does not require source domain data at the meta-level. Its meta-classifier requires little hyperparameter tuning. FES is also computationally cheaper, unless the number of few-shot learning tasks is very large, e.g., in the thousands, where the total cost of performing FES on all tasks begins to exceed that of training a universal model once. Therefore, to field practitioners who wish to use backbones and fine-tuning algorithms specific to their work, FES is likely more flexible and user-friendly than universal-model methods.

10 Declarations

- Funding: This research is funded by the Ministry of Business, Innovation and Employment of New Zealand as part of a Smart Ideas project entitled “User-friendly Deep Learning”, please refer to https://www.mbie.govt.nz/science-and-technology/science-and-innovation/funding-information-and-opportunities/investment-funds/endeavour-fund/.
- Conflict of interest/Competing interests: On behalf of all authors, the corresponding author states that there is no conflict of interest.
- Ethics approval: Not applicable
- Consent to participate: Not applicable
- Consent for publication: Not applicable
- Availability of data and materials: All data used can be acquired publicly via https://github.com/google-research/meta-dataset for the official Meta-Dataset, https://github.com/cambridge-mlg/cnaps for three additional target domains, https://github.com/IBM/cdfs-l-benchmark for four additional target domains, and https://data.vision.ee.ethz.ch/cvl/datasets_extra/food-101/ for one additional target domain.
- Code availability: The implementation and the computational work are done using the Python programming language and the PyTorch deep learning library (Paszke et al., 2019). The code and data files are available via GitHub at https://github.com/HongyuJerryWang/FeatureExtractorStacking.
• Authors’ contributions: All authors contributed to the study conception and design. Material preparation and data collection and analysis were performed by Hongyu Wang. The first draft of the manuscript was written by Wang and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

• Human and Animal Ethics: Not applicable

References

Bateni, P., Barber, J., van de Meent, J., Wood, F. (2022). Enhancing few-shot image classification with unlabelled examples. *IEEE/CVF Winter Conference on Applications of Computer Vision, Waikoloa, HI, USA* (pp. 1597–1606). IEEE.

Bateni, P., Goyal, R., Masrani, V., Wood, F., Sigal, L. (2020). Improved few-shot visual classification. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA* (pp. 14481–14490). Computer Vision Foundation / IEEE.

Bossard, L., Guillaumin, M., Gool, L.V. (2014). Food-101 - mining discriminative components with random forests. *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland* (Vol. 8694, pp. 446–461). Springer.

Chen, W., Liu, Y., Kira, Z., Wang, Y.F., Huang, J. (2019). A closer look at few-shot classification. *7th International Conference on Learning Representations, New Orleans, LA, USA*. OpenReview.net.

Chen, Y., Liu, Z., Xu, H., Darrell, T., Wang, X. (2021). Meta-baseline: Exploring simple meta-learning for few-shot learning. *2021 IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada* (pp. 9042–9051). IEEE.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Miami, Florida, USA* (pp. 248–255). IEEE Computer Society.

Dvornik, N., Schmid, C., Mairal, J. (2020). Selecting relevant features from a multi-domain representation for few-shot classification. *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK* (Vol. 12355, pp. 769–786). Springer.
Galeano, P., Joseph, E., Lillo, R.E. (2015). The Mahalanobis distance for functional data with applications to classification. *Technometrics, 57*(2), 281–291.

Guo, Y., Codella, N., Karlinsky, L., Codella, J.V., Smith, J.R., Saenko, K., ... Feris, R. (2020). A broader study of cross-domain few-shot learning. *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK* (Vol. 12372, pp. 124–141). Springer.

He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA* (pp. 770–778). IEEE Computer Society.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of the 32nd International Conference on Machine Learning, Lille, France* (Vol. 37, pp. 448–456). JMLR.org.

Li, W., Liu, X., Bilen, H. (2021). Universal representation learning from multiple domains for few-shot classification. *2021 IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada* (pp. 9506–9515). IEEE.

Li, W., Liu, X., Bilen, H. (2022). Cross-domain few-shot learning with task-specific adapters. *IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA* (pp. 7151–7160). IEEE.

Li, W.-h., Liu, X., Bilen, H. (2022). Universal representation learning and task-specific adaptation for few-shot learning. https://github.com/VICO-UoE/URL. (Accessed: 2022-09-29)

Liu, L., Hamilton, W.L., Long, G., Jiang, J., Larochelle, H. (2021). A universal representation transformer layer for few-shot image classification. *9th International Conference on Learning Representations, Virtual Event, Austria*. OpenReview.net.

Liu, Y., Lee, J., Zhu, L., Chen, L., Shi, H., Yang, Y. (2021). A multi-mode modulator for multi-domain few-shot classification. *2021 IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada* (pp. 8433–8442). IEEE.

Mensink, T., Verbeek, J., Perronnin, F., Csurka, G. (2013). Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35*(11), 2624–2637.
Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems 32, Vancouver, BC, Canada (pp. 8024–8035).

Perez, E., Strub, F., de Vries, H., Dumoulin, V., Courville, A.C. (2018). FiLM: Visual reasoning with a general conditioning layer. Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, the 30th innovative Applications of Artificial Intelligence, and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence, New Orleans, Louisiana, USA (pp. 3942–3951). AAAI Press.

Requeima, J., Gordon, J., Bronskill, J., Nowozin, S., Turner, R.E. (2019). Fast and flexible multi-task classification using conditional neural adaptive processes. Advances in Neural Information Processing Systems 32, Vancouver, BC, Canada (pp. 7957–7968).

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 115(3), 211–252.

Snell, J., Swersky, K., Zemel, R.S. (2017). Prototypical networks for few-shot learning. Advances in Neural Information Processing Systems 30, Long Beach, CA, USA (pp. 4077–4087).

Tan, M., & Le, Q.V. (2021). Efficientnetv2: Smaller models and faster training. Proceedings of the 38th International Conference on Machine Learning, Virtual Event (Vol. 139, pp. 10096–10106). PMLR.

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., Knight, K. (2005). Sparsity and smoothness via the fused lasso. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(1), 91–108.

Triantafillou, E., Larochelle, H., Zemel, R.S., Dumoulin, V. (2021). Learning a universal template for few-shot dataset generalization. Proceedings of the 38th International Conference on Machine Learning, Virtual Event (Vol. 139, pp. 10424–10433). PMLR.

Triantafillou, E., Zhu, T., Dumoulin, V., Lamblin, P., Evci, U., Xu, K., ... Larochelle, H. (2020). Meta-dataset: A dataset of datasets for learning to learn from few examples. 8th International Conference on Learning Representations, Addis Ababa, Ethiopia. OpenReview.net.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., ... Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems 30, Long Beach, CA, USA* (pp. 5998–6008).

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., Wierstra, D. (2016). Matching networks for one shot learning. *Advances in Neural Information Processing Systems 29, Barcelona, Spain* (pp. 3630–3638).

Wolpert, D.H. (1992). Stacked generalization. *Neural Networks, 5*(2), 241–259.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Póczos, B., Salakhutdinov, R., Smola, A.J. (2017). Deep sets. *Advances in Neural Information Processing Systems 30, Long Beach, CA, USA* (pp. 3391–3401).

**A Additional heatmaps**

Additional heatmaps visualising kernel weights on target domains with TSA fine-tuning are shown by Figures 14-40.

---

**Fig. 14:** FES kernel for mscoco

**Fig. 15:** ConFES kernel for mscoco

**Fig. 16:** ReFES kernel for mscoco
Fig. 17: FES kernel for mnist

Fig. 18: ConFES kernel for mnist

Fig. 19: ReFES kernel for mnist

Fig. 20: FES kernel for cifar10

Fig. 21: ConFES kernel for cifar10

Fig. 22: ReFES kernel for cifar10
Fig. 23: FES kernel for cifar100

Fig. 24: ConFES kernel for cifar100

Fig. 25: ReFES kernel for cifar100

Fig. 26: FES kernel for CropDisease

Fig. 27: ConFES kernel for CropDisease

Fig. 28: ReFES kernel for CropDisease
Fig. 29: FES kernel for EuroSAT

Fig. 30: ConFES kernel for EuroSAT

Fig. 31: ReFES kernel for EuroSAT

Fig. 32: FES kernel for ISIC

Fig. 33: ConFES kernel for ISIC

Fig. 34: ReFES kernel for ISIC
Fig. 35: FES kernel for ChestX

Fig. 36: ConFES kernel for ChestX

Fig. 37: ReFES kernel for ChestX

Fig. 38: FES kernel for Food101

Fig. 39: ConFES kernel for Food101

Fig. 40: ReFES kernel for Food101
FES for CDFSML

B Information sheet

Please find our answers to the information sheet questions below.

Q: What is the main claim of the paper? Why is this an important contribution to the machine learning literature?
A: We propose a novel cross-domain few-shot meta-learning (CDFSML) method termed Feature Extractor Stacking (FES), inspired by stacked generalisation, and two FES variants: Convolutional FES (ConFES) and Regularised FES (ReFES). FES and its variants are highly competitive CDFSML methods and have certain practical advantages in real-world scenarios.

Q: What is the evidence you provide to support your claim?
A: We evaluate FES and its variants on the well-known Meta-Dataset benchmark for CDFSML and show that they advance the state of the art on the benchmark, outperforming recent methods in the literature, including TSA, URL, and FLUTE.

These recent methods rely on a universal model, which is slow to train and may be incompatible with heterogeneous base models. On the other hand, FES fine-tunes individual base models directly on a few-shot support set, making it fast to perform and compatible with heterogeneous base models.

Q: What papers by other authors make the most closely related contributions, and how is your paper related to them?
A: We use the Meta-Dataset benchmark:
Triantafillou, E., Zhu, T., Dumoulin, V., Lamblin, P., Evci, U., Xu, K., ... Larochelle, H. (2020). Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples. 8th International Conference on Learning Representations, Addis Ababa, Ethiopia. OpenReview.net.

We compare to three methods in recent CDFSML literature:
TSA: Li, W.-H., Liu, X., & Bilen, H. (2022). Cross-domain Few-shot Learning with Task-specific Adapters. IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 7151–7160. IEEE.
URL: Li, W.-H., Liu, X., & Bilen, H. (2021). Universal Representation Learning from Multiple Domains for Few-shot Classification. 2021 IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 9506–9515. IEEE.
FLUTE: Triantafillou, E., Larochelle, H., Zemel, R. S., & Dumoulin, V. (2021). Learning a Universal Template for Few-shot Dataset Generalization. Proceedings of the 38th International Conference on Machine Learning, Virtual Event, 139, 10424–10433. PMLR.

Q: Have you published parts of your paper before, for instance in a conference?
A: No.