Meta-Album: Multi-domain Benchmark for Few-Shot Image Classification

Ihsan Ullah[*], Phan Anh Vu[*], Haozhe Sun[*], Mike Huisman[+], Felix Mohr[‡], Jan N. van Rijn[+], Aske Plaat[+], Adrian El Baz[†], Zhengying Liu[*], Joaquin Vanschoren[§], Isabelle Guyon[*†]
∗ LISN (CNRS/INRIA) Université Paris-Saclay, France
† ChaLearn, California, USA
+ LIACS, Leiden University, The Netherlands
‡ Universidad de La Sabana, Colombia
§ Eindhoven University of Technology, The Netherlands
meta-album@chalearn.org

Abstract

We introduce a new benchmark for few-shot learning, meta-learning and other computer vision tasks, consisting of 15 datasets from 5 domains: ecology, biomedicine, manufacturing, optical character recognition, and remote sensing. The datasets will be used in the NeurIPS MetaDL Challenge 2021. They are image classification datasets, uniformly formatted as 128x128 RGB images, carefully resized with anti-aliasing, cropped manually, and annotated with various meta-data, including super-classes. We provide data quality control software and means of creating tasks for various scenarios (either research or application oriented). We also provide the source code of baseline methods and instructions for contributing either new datasets or algorithms to our expandable benchmark. All 15 datasets will be made available in a controlled environment where new algorithms can be evaluated with free computational resources, and 10 training datasets will be publicly released. We showcase the utility of these datasets in the context of various use cases, including large-scale multi-class classification, few-shot learning, and transfer learning. Compared with other existing benchmarks, our work fills an important gap between single-domain small-scale benchmarks, and multi-domain large-scale benchmarks. The small-scale datasets, such as "Omniglot" and "miniImageNet", usually have at most 70,000 images in total, weigh at most a few GB, and can be run on regular machines. The large-scale benchmarks, such as "Meta-dataset" and "VTAB", have more than 50 million images, weigh at least a few hundreds GB, and require high-end super-computer clusters. In contrast, our benchmark includes around 120,000 images, weighs roughly 2.3 GB, and is amenable to run in a few hours on regular GPUs.

1 Introduction

Machine learning has progressed rapidly in recent years and has enabled breakthroughs in various domains. The success of most machine learning techniques hinges on the availability of large amounts of data [48, 26], limiting their applicability in domains where only little data is available. Enabling machine learning algorithms to learn new tasks from only a few examples is studied within the field of *few-shot learning* [31, 42, 50]. Novel meta-learning algorithms have recently been proposed targeting few-shot learning, triggering a surge of popularity for such problems [57, 19, 17].

Despite the popularity of the field, progress is held back by a lack of good, challenging, and computationally feasible benchmarks, that enable us to accurately assess the generalization abilities of

Preprint. Under review.

few-shot learning algorithms. To remedy this, we introduce the **Meta-Album** benchmark, consisting of 15 image classification datasets from different domains. It includes around 120,000 images and weighs roughly 2.3 GB, making it amenable to training models in a few hours on regular GPUs. It is thus positioned between small-scale datasets such as Omniglot [24], miniImageNet [54, 37] and CUB [55], which usually have at most 70,000 images in total and weigh at most a few GB, and very large-scale benchmarks such as Meta-dataset [51] and VTAB [8] which have more than 50 million images, weigh at least a few hundreds GB, and require high-end super-computer clusters. For comparison, we highlight the features of Meta-Album and previous benchmarks in Table 1 and provide details in Appendix J. Meta-Album aims at facilitating algorithm comparisons, particularly across domains, by formatting all images with the same size. To that end, we optimized cropping and resizing to reduce dimensions as much as possible without degrading performance too much.

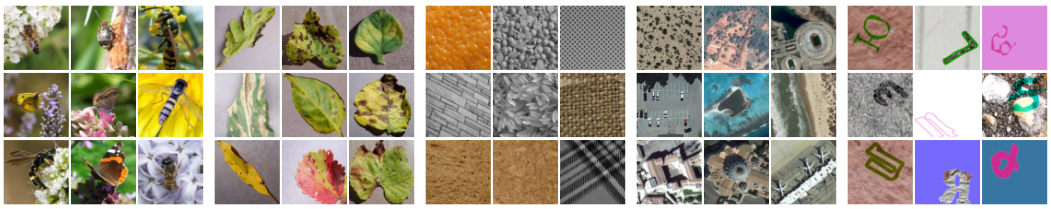
Furthermore, we also introduce a more realistic evaluation approach. Traditionally, few-shot learning algorithms have been evaluated by taking an existing benchmark dataset from a particular domain (*e.g.,* handwriting recognition) with a large number of classes, and then breaking it down into classification sub-tasks, each including a *random* subset of classes (*e.g.,* a few specific characters). Algorithms are then tested for their ability to solve such sub-tasks "quickly" from a small *fixed* number of examples, after being trained on many other sub-tasks. With n classes and k examples, this is called *n-way, k-shot* learning. While this setting has served research well, it is not very representative of practical real-world applications, where new (sub)tasks may come from multiple sources or domains, and are not drawn at random, but derived from a class hierarchy. To facilitate benchmarking in a more realistic setting, we provide data formatted in a uniform way (images of a given size) but coming from a variety of domains. Furthermore, we use a hierarchy of natural classes and super-classes to facilitate creating more natural sub-tasks i.e. enabling the creation of tasks with variable numbers of ways and shots.

Table 1: Feature comparison between Meta-Album and other benchmarks/datasets

Benchmark/ Dataset	# of domains	# of images	min/max classes per domain	min/max images per class	size on disk	multi-domain	reasonable size	uniform # of images per class	uniform image size	extendable
Meta-Dataset	7	53,068,000	43/1,696	3/140,000	210 GB	✓	✗	✗	✗	✗
VTAB	3	2,244,000	2/397	40/1000	100 GB	✓	✗	✗	✗	✗
MS-COCO	1	328,000	80/80	9/10,777	44 GB	✗	✗	✗	✗	✗
Mini Imagenet	1	60,000	100/100	600/600	1 GB	✗	✗	✓	✓	✗
Omniglot	1	32,000	1623/1623	20/20	148 MB	✗	✓	✓	✓	✗
CUB-200	1	6,000	200/200	20/39	647 MB	✗	✓	✗	✗	✗
CIFAR-100	3	60,000	15/50	600/600	161 MB	✗	✓	✓	✓	✗
Meta-Album	**5**	**120,000**	**27/706**	**40/40**	**2.3 GB**	✓	✓	✓	✓	✓

The **Meta-Album** benchmark is illustrated in Figure 1. We preprocessed and formatted 15 datasets spread uniformly across 5 different domains (ecology, bio-medicine, manufacturing, optical character recognition, and remote sensing). Some of these datasets have not previously been published, while others are public datasets re-purposed here to provide better domain coverage. Ten of these have been used as part of the MetaDL 2021 challenge, an official NeurIPS competition. We will publicly release 10 out of the 15 datasets after the NeurIPS 2021 MetaDL challenge. The remainder is used for blind testing purposes. To perform such blind testing, foster reproducible science, and democratize the use of our proposed benchmark, the datasets will be made available through a benchmark platform allowing users to submit and execute their code in a free controlled environment: the Codabench platform. Participants can choose to make their code publicly available on the platform.

Additionally, **Meta-Album** has been designed to be expandable: we propose a protocol to stimulate continual community-based growth and development of the benchmark, and encourage researchers in the field to upload new benchmarks or algorithms. We do this by publishing detailed instructions regarding the data formatting and processing pipeline, together with data quality control software in the dedicated GitHub repository. Such datasets will be periodically reviewed by our benchmark



(a) Insects (c) Mini Plant Village (e) Textures (g) Mini RESISC (i) OmniPrint-MD-mix

(b) Plankton (d) Medleaf (f) Texture DTD (h) Mini RSICB (j) OmniPrint-MD-5-bis

Figure 1: Meta-Album public datasets sample images.

committee, and eventually incorporated in our benchmark suite, if they meet the same criteria on quality and level of documentation as the datasets described in this paper.

In short, the contributions of our work are the following.

- We provide a new few-shot benchmark consisting of 15 uniformly formatted data sets from 5 domains, which facilitates cross-domain meta-learning and practical and realistic evaluation of few-shot algorithms. Ten of these datasets will be made public after the NeurIPS 2021 MetaDL benchmark end, and the remaining five within 2 years of the termination of the challenge.
- When data are released, in addition to formatted data, the original raw data will be released to facilitate further research.(https://github.com/ihsanullah2131/meta-album/tree/master/BenchmarkDatasets)
- We provide a controlled environment with free available computational resources to evaluate new algorithms on all benchmark datasets through code submission.
- We stimulate community-driven growth by providing software and instructions to create additional benchmark datasets, and a strict quality-control and review process (https://github.com/ihsanullah2131/meta-album).
- We showcase our new benchmark by performing an experimental evaluation on a number of use cases, including transfer and meta-learning tasks, using a variety of algorithms, for which we release the code.

2 Related work

In this section, we review previously proposed benchmarks for few-shot learning, meta-learning, as well as large-scale multi-domain benchmarks.

Single dataset benchmark: Omniglot [24] is often used as a starting benchmark for few-shot and meta-learning. MiniImageNet [54] and Tiered-ImageNet [39] are adapted for few-shot image classification from ImageNet [41]. CIFAR-FS [1] and FC100 [32] are remodeled from CIFAR-100 [22] for few-shot settings.

Multi-dataset benchmark: A recent trend is to assemble numerous datasets from different domains in the same benchmark. Visual Decathlon [38] gathers 10 diverse datasets. The focus is finding a model with universal representation capacity for use in many tasks. VTAB (Visual Task Adaptation

Benchmark) [61] assembles 19 image classifications task across various domain. These tasks are grouped into 3 partitions: natural, specialized, and structured. Meta-Dataset [51] includes 10 image classification datasets from several application domains in one collection. Meta-Dataset also leverages the label hierarchy in ImageNet and Omniglot to organize the tasks.

Transfer learning and meta-learning benchmarks: VTAB + MD [8] attempts to unify common transfer and meta-learning datasets in a single benchmark. The authors also provide a comparison of popular meta and transfer learning methods. BSCD-FSL (Broader Study of Cross-Domain Few-Shot Learning) [13] gathers 4 real-world tasks to compare few-shot, meta-learning and transfer learning methods. WILDS [21] is a benchmark of 10 datasets with multiple modalities (including images, graphs, and text), reflecting a diverse range of distribution shifts that naturally arise in real-world applications, and hence useful to evaluate meta-learning, and transfer learning techniques. In reinforcement learning, sets of simulation environments exist for meta- and transfer learning, such as Meta-World [60].

Outside few-shot and meta-learning: The AutoDL challenge [28] releases a series of 66 datasets from numerous domains. These datasets cover a wide range of modalities: image, video, audio, text, tabular. This competition focuses on finding a universal algorithm, which can solve many tasks without human supervision.

Alongside these works, our dataset is complementary to previously proposed benchmarks. While building on public datasets to vary the domains, we also introduce multiple freshly curated datasets. We aim at including various domains, while keeping the size portable. Specifically, the Insects dataset is introduced to the public for the first time. For the secret bio-medicine dataset, we provided additional bounding box annotation in addition to the original classification labels.

3 Problem formulation

Our benchmark is situated in the setting of multi-class image classification. Here, a *learning task* $\mathcal{T}$ consists of an *instance space* $\mathcal{X}$, which in our case are RGB images of size 128×128, and a posterior (unknown) probability distribution on a *label space* $\mathcal{Y}$, i.e. $\mathbb{P}_\mathcal{T}(\mathcal{Y} \mid x)$ for $x \in \mathcal{X}$. The goal of standard supervised learning is to find, given a finite i.i.d. dataset $D_\mathcal{T} \sim \mathbb{P}_\mathcal{T}$, a hypothesis $h : \mathcal{X} \to \mathcal{Y}$ that minimizes the expected task loss $loss_\mathcal{T}$ over the set of *all* possible instances that come from $\mathbb{P}_\mathcal{T}$, i.e., $\mathcal{R}_\mathcal{T}(h) = \mathbb{E}_{(x_i,y_i)\sim\mathbb{P}_\mathcal{T}} loss_\mathcal{T}(y_i, h(x_i))$. This is often called the *risk* of an hypothesis h for task $\mathcal{T}$. In *few-shot learning*, the dataset $D_\mathcal{T}$ from which we can learn a given task consists of only a few examples.

One of the most common few-shot learning problem setups is called N-way k-shot classification. In this setting, a learning algorithm $\mathcal{A}$ is presented with a support set $D_\mathcal{T}^{tr} \sim \mathbb{P}_\mathcal{T}$. A support set consists of k examples for each out of N classes (for small k). The success of the learning algorithm $\mathcal{A}$ is then measured on the query set $D_\mathcal{T}^{te}$ which contains unseen examples from task $\mathcal{T}$. Importantly, the quality of a learning algorithm $\mathcal{A}$ should be measured by tasks containing unseen classes.

Meta-learning is a popular approach to the few-shot learning problem. The key idea behind this approach is to produce a learning algorithm $\mathcal{A}$ that is capable of learning new tasks from limited data [17, 19]. Since the concrete task is unknown at the time where the learner is created, it should work well on average over *all* possible tasks. So the goal of meta-learning, in the context of N-way k-shot classification, is to minimize

$$\mathcal{L}(\mathcal{A}) = \mathbb{E}_{\mathcal{T}\sim\mathbb{P}', D_\mathcal{T}^{tr}} \left[\mathcal{R}_\mathcal{T}(\mathcal{A}(D_\mathcal{T}^{tr})) \right], \tag{1}$$

where $\mathcal{A}$ is a parameterized learning algorithm, $\mathbb{P}'$ is a distribution over N-class tasks, $\mathcal{R}_\mathcal{T}$ is the risk function of task $\mathcal{T}$, and $D_\mathcal{T}^{tr} \sim \mathbb{P}_\mathcal{T}$ is the support set sampled from the task distribution $\mathbb{P}_\mathcal{T}$.

Since neither $\mathcal{R}_\mathcal{T}$ let alone $\mathcal{L}$ can be computed in practice, our benchmark follows a common routine that uses the given data to *estimate* the score in Eq. (1). Let D be the full dataset to which we have access. We partition the set of *labels* $\mathcal{Y}$ occurring in the given data D into $\mathcal{Y}^{meta\text{-}train}$ and $\mathcal{Y}^{meta\text{-}test}$, and the dataset D is partitioned accordingly into $D^{meta\text{-}train}$ and $D^{meta\text{-}test}$. The meta-training dataset $D^{meta\text{-}train}$ is then used to induce a learner $\mathcal{A}$, and $D^{meta\text{-}test}$ is used to estimate the quality of the learner on new tasks containing unseen classes. Formally, the benchmark assesses the performance of a meta-learner $\mathcal{A}_{meta}$ by letting it produce a learner $\mathcal{A} = \mathcal{A}_{meta}(D^{meta\text{-}train})$, then averaging the empirical score of $\mathcal{A}$ over J tasks $\mathcal{T}$ sampled from $D^{meta\text{-}test}$. The *estimated* performance of the meta-learner is then

$$\frac{1}{J}\sum_{j=1}^{J}\frac{1}{|D^{te}_{\mathcal{T}_j}|}\sum_{(x_i,y_i)\in D^{te}_{\mathcal{T}_j}} loss_{\mathcal{T}}(y_i, h_j(x_i)), \tag{2}$$

where $D^{te}_{\mathcal{T}_j} \subseteq D^{meta\text{-}test}$ is the sampled query set of task $\mathcal{T}_j$ and $h_j := \mathcal{A}_{meta}(D^{meta\text{-}train})(D^{tr}_{\mathcal{T}_j})$ is the hypothesis produced by deploying the learner (obtained from meta-training on $D^{meta\text{-}train}$) on the support set $D^{tr}_{\mathcal{T}_j}$. $loss(y, \hat{y})$ is the penalty received when predicting $\hat{y}$ when the correct answer is y. It would also be possible to average over different splits of $\mathcal{Y}^{tr}$ and $\mathcal{Y}^{te}$, but this would also imply performing meta-training several times (with different random seeds), which would be computationally more demanding.

4 Design of datasets

We performed several iterations of preprocessing, experiments and analysis to prepare the datasets. This workflow includes: identifying bias in the data, identifying artifacts in the images, and make sure the images are recognizable by human eyes.

4.1 Data preparation

As we work with datasets from diverse sources, each dataset in this benchmark may require different preprocessing strategies, e.g., the insects dataset, plant-diseases dataset, manufacturing, and remote sensing datasets have images in different resolutions and orientations, and usually, the object of interest lies in the middle of the image. The images in these datasets are cropped horizontally or vertically to get a perfect squared image. In the plankton dataset, the image orientation depends on the shape of the plankton and the way it is photographed, i.e., images have vertical or horizontal orientations based on the plankton in the image. Cropping images is not useful in this case because a big part of the plankton would be cropped. As such, we have added a matching background to the images either horizontally or vertically by extending the top and bottom 3 rows or left and right columns respectively and applying Gaussian kernel of size (29, 29) using open-cv [2] to the newly constructed background to make sure that we do not introduce artifacts in the data. For all datasets except for the optical character recognition datasets, we resize the images to a 128x128 resolution using an anti-aliasing filter [2]. The optical character recognition datasets are synthesized by OmniPrint [49] (MIT license) and do not need further processing.

The preprocessed data is formatted in a defined Data format. Only classes with at least 40 examples are kept for each dataset to maintain a balance between a large number of classes and sufficient examples per class. Any additional examples per class are removed. In this way, we balance the datasets while keeping them compact. More details about the data preparation and data format can be found in Appendix A.

4.2 Quality control

In order to ensure the high quality of datasets, we put a strong emphasis on quality control. In the next sections, we describe two ways to ensure this quality: factsheets and GradCam visualizations.

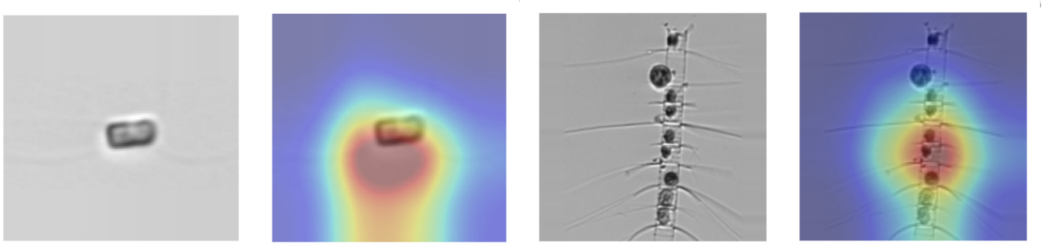


Figure 2: GradCAM activation maps to detect artifacts that could be introduced by image preprocessing.

Factsheet

For each dataset, we contribute a factsheet, an automatically generated report showing basic information and experimental results on the dataset. These experiments are run on the formatted datasets, i.e, datasets with 40 images per class, processed according to the procedure described in Section 4.1. The experiments are flexible and the configuration can be changed easily by changing the parameters of the software. For the datasets formatted in this paper, we train a randomly initialized ResNet-18 architecture [15] on various 5-way 20-shot classification tasks. That is, the network is presented with support sets consisting of 20 images for each of the 5 classes. The remaining 20 images per class are used in the query set to measure the generalization ability of the network. Note that the network trains from scratch (randomly initialized weights) on every task. From the experiments, we construct ROC curves for every task for all classes. Classes that are too hard to separate (indicating non-learnable tasks) with the network can be removed from the data, after also checking the performance of a pre-trained ResNet-18 on ImageNet. If in both cases the area under the curve (AUC score) for a given class is below 0.7 (slightly better than random performance), it is discarded from the dataset.

In addition to this automatic quality check, we also generate Imagesheets for the datasets, which are PDF files with images of every class. This way, we can also check the data quality visually. We provide more details about the data format and the Factsheets in Appendix B.

GradCam visualization

In order to detect potential bias stemming from the data and our curating process, we inspect where the baseline model focuses with GradCAM [43]. GradCAM is a post-hoc attention method, showing the contribution of each input element to the output.

By displaying the activation map on top of the original input image, we can observe the impact of each pixel on the prediction output. As such, we can verify whether the model focuses on the relevant subject. In this way, it can also be used to detect whether artifacts are introduced by the image processing in the picture or on the background and other artifacts. For example, we use GradCAM for detecting any artifacts introduced by preprocessing the backgrounds of plankton images to make squared images (Figure 2). Other method should also be considered to diagnose data [62].

5 Datasets

Meta-Album consists of 3 datasets for each of the following 5 domains: ecology, bio-medicine, manufacturing, remote sensing, and optical character recognition. All datasets have a variety of categories/classes and there are exactly 40 images per category. These datasets also come with a variety of meta-data. All 15 datasets are chosen after careful and critical analysis during the data preparation and quality control steps as described in Section 4. Table 2 summarizes the public datasets; Figure 1 shows sample images from each public dataset.

The public datasets from ecology domain consists of Insects (pollinating insects i.e. bees,wasps, butterflies etc) and Plankton (microscopic sea organisms) datasets. Bio-medicine domain has two public datasets: Mini Plant Village (healthy and sick leaves) and Medleaf (healthy medicinal leaves). The public manufacturing datasets have images of textures/patterns of different natural surfaces. The remote-sensing domain has two public aerial datasets which consists of satellite images. The public datasets from OCR domain consist of OmniPrint-MD-5-bis and OmniPrint-MD-mix. The first consists of color text on natural backgrounds, the second contains images synthesized with 20 different styles (hollow text, color text, text with outline, different outline filling, etc.). The set of backgrounds and synthesis parameters are also different between OmniPrint-MD-5-bis and OmniPrint-MD-mix. More details about datasets and their meta-data are listed in Appendix C.

The Meta-Album datasets will be used in the NeurIPS MetaDL Challenge 2021. After the competition is concluded, all 15 datasets will be available on the Codabench platform [58]. Codabench is a platform that facilitates benchmarking, by running submissions of the participants on the data in a controlled reproducible environment (implemented with dockers), with an organizer-designed protocol. The 10 public benchmark datasets will be downloadable, as well as baseline algorithms, accessible through Codabench. But, the 5 private datasets will remain hidden until a later release, for the purpose of blind-testing algorithms on the platform. We intend to implement a rolling benchmark to keep exposing the community to fresh data and alleviate habituation/overfitting problems. After a

Table 2: Meta-Album: Public datasets summary

Domain	Dataset	Categories	Images	Original source
Ecology	Insects	114	4,560	SPIPOLL [44]
	Plankton	91	3,640	WHOI [16]
Bio-medicine	Mini Plant Village	37	1,480	Plant Village [18, 33]
	Medleaf	27	1,080	Medicinal Leaf [40]
Manufacturing	Textures	64	2,560	KTH-TIPS [10, 30] Kylberg [23] UIUC [25]
	Texture DTD	47	1,880	Texture DTD [6]
Remote sensing	Mini RESISC	45	1,800	RESISC45 [5]
	Mini RSICB	45	1,800	RSICB128 [27]
OCR	OmniPrint-MD-mix	706	28,240	OmniPrint [49]
	OmniPrint-MD-5-bis	706	28,240	OmniPrint [49]

while, private datasets are released and replaced by new private datasets. For details about private datasets, see Appendix D. License information for all datasets can be found in Appendix E.

Details about how to access the benchmark datasets, contribute to the open benchmark, prepare new datasets with quality control, and how to submit these datasets to the benchmark can be found on the Meta-Album GitHub repository. All the datasets will remain static (unless an update is required) and regular backups will be facilitated by Codabench. In any case, regular updates will be available on the dedicated GitHub repository.

6 Use Cases and Baselines

In this section, we showcase how the benchmark infrastructure can be used on different problem types. Besides few-shot learning, for which it has been initially designed, it can be used for regular multiclass image classification, transfer learning, and hierarchical classification. In the following, we explain for each of these use cases how the benchmark can be used and also provide results of typical baselines on each of them.

Prototypical Network is the best overall model. Surprisingly, Fine-tuning is quite competitive with this best approach. One remarkable exception is the OmniPrint datasets from OCR domain, where Prototypical Network outperforms all other methods by a large margin. OmniPrint datasets [49] are synthetic datasets generated by software, and are very different from the source domain (Imagenet). Meta-learning algorithms seem to have an edge in this case.

6.1 Few-shot learning

For few-shot-learning, we investigate the 5-way-5-shot performance of popular meta-learning baselines on all 15 datasets; this is the problem described in Section 3 and the one addressed in the NeurIPS MetaDL Challenge 2021. To this end, we use the ResNet-18 backbone [15] and use the best-reported hyperparameters by the original authors on 5-way-5-shot miniImageNet. For a given dataset, we train all techniques on 80,000 tasks and validate them every 2,500 tasks. Note that this results in 15 models per dataset. The query set for every task contains 16 examples per class, following [3]. The learning algorithm with the best validation performance is evaluated on 600 randomly sampled meta-test tasks. We average the results over 3 runs with different random seeds. For all experiments, we use PNY GeForce RTX 2080TI GPUs with 11GB of VRAM.

The considered baselines are the following. **TrainFromScratch** starts with randomly initialized parameters when learning a new task. **Finetuning** pre-trains on all data in $D^{meta\text{-}train}$, freezes all hidden layers and trains a new output layer on every task at test time. **MAML** [9] learns a set of initialization parameters from which it can learn new tasks quickly (within a few gradient update steps on the support set). We use first-order MAML with an inner learning rate of $\alpha = 0.01$ and an outer learning rate of $\beta = 0.001$. **Matching networks** [54] learn a feature space that allows making

Table 3: **Few-shot learning use case.** The average meta-test accuracy scores (%) averaged over 3 different runs. The $\pm x$ denote the 95% confidence intervals. Bold entries denote the best performance per dataset. A lower average rank indicates better relative performance. Used abbreviation: "TFS": TrainFromScratch, "2h30m": 2 hours and 30 minutes.

Domain	Dataset	TFS	Finetuning	MAML	MatchingNet	ProtoNet
Ecology	Insects	26.6±0.5	**50.9±0.7**	40.0±0.7	43.1±0.7	47.0±0.7
	Plankton	54.8±0.8	**73.2±0.7**	70.0±0.8	70.1±0.8	73.0±0.8
	Hidden Ecology	49.5±0.8	73.6±0.7	69.4±0.8	65.9±0.7	**74.8±0.6**
Bio-medicine	Mini Plant Village	59.0±0.7	82.0±0.5	68.3±0.6	69.3±0.5	**83.3±0.4**
	Medleaf	58.5±0.7	69.6±0.6	68.3±0.6	52.6±0.6	**71.1±0.5**
	Hidden Bio-Medicine	45.3±0.7	55.1±0.7	50.6±0.7	51.0±0.7	**59.2±0.7**
Manufacturing	Textures	71.0±1.0	**93.0±0.4**	89.8±0.6	88.8±0.5	**93.0±0.4**
	Texture DTD	28.9±0.5	**38.0±0.5**	36.9±0.5	32.1±0.5	35.6±0.5
	Hidden Manufacturing	71.0±1.0	**93.0±0.4**	89.8±0.6	88.8±0.5	**93.0±0.4**
Remote sensing	Mini RESISC	43.0±0.7	**59.6±0.7**	58.0±0.6	48.6±0.5	59.3±0.6
	Mini RSICB	58.9±0.8	**86.9±0.5**	81.4±0.6	74.0±0.6	84.0±0.5
	Hidden Remote Sensing	29.6±0.5	**40.4±0.7**	40.3±0.6	39.6±0.7	39.6±0.6
OCR	OmniPrint-MD-mix	19.8±0.3	56.3±0.7	20.0±0.3	82.9±0.7	**96.0±0.3**
	OmniPrint-MD-5-bis	20.4±0.3	62.5±0.8	20.5±0.4	90.7±0.5	**96.4±0.3**
	Hidden OCR	22.0±0.4	94.3±0.3	21.1±0.4	90.9±0.5	**97.1±0.2**
Average rank		4.9	1.8	3.4	3.4	1.5
Average run-time		30m	2h30m	4h20m	2h10m	2h10m

predictions using pair-wise distances between query inputs and support examples. **Prototypical networks** [45] use the same strategy but base predictions on distances of query inputs to class prototypes computed from the support sets. Importantly, these techniques start with randomly initialized weights and do not rely on pre-trained weights on, e.g., ImageNet [41].

Table 3 displays the results. As we can see, the datasets differ in the difficulty of associated tasks. Nonetheless, every dataset presents tasks containing learnable concepts since, for every dataset, there is at least one technique yielding better performance than a majority class baseline (20% accuracy for a 5-way problem). Overall, prototypical networks yield the best average performance across datasets (corresponding to the lowest rank), while also being one of the fastest algorithms.

6.2 Transfer learning: fine-tuning a pre-trained network

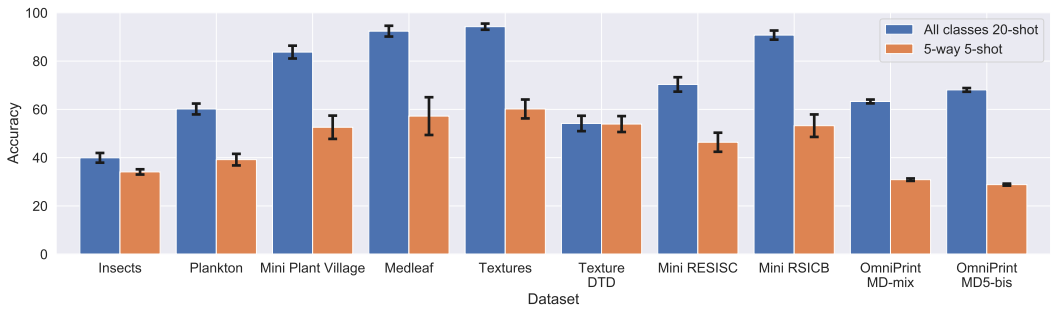


Figure 3: **Transfer learning by fine-tuning a pre-trained network:** Average accuracy scores for the fine-tuning (pretrained on ImageNet) technique in the all classes 20-shot and 5-way 5-shot settings. The error bars indicate the 95% confidence intervals.

Our benchmark datasets can also be used for transfer learning. To illustrate this, we finetune a pre-trained ResNet-18 [15] on the support sets of different randomly sampled tasks. The learning rate of the first layers is set small (10^{-7} and increases following a geometric progression to 10^{-3} in the output layer). We average the performances over 3 different runs for two scenarios: all-way 20-shot (all classes) and 5-way 5-shot using 20 and 35 query examples per class, respectively. The results of

are shown in Figure 3. Clearly, there is better transfer to some datasets (*e.g.,* Textures) than others (*e.g.,* Insects). As expected, the 5-way 5-shot results by pretraining on the meta-train set (column Finetuning of Table 3) are generally better than when pretraining on ImageNet (column 5-way 5-shot of Table 5 in Appendix F). The difficulty of transferring knowledge can be adjusted by decreasing the number of classes and the number of examples per class. More details about experiment setup, result and error bar can be found in Appendix F.

6.3 Hierarchical classification

Meta-Album consists of five datasets that contain hierarchical classes, i.e., they contain regular classes and super classes. This hierarchical structure can be used to perform fine-grained classification within super classes. One such task, corresponding to a single super class, is constructed as follows. We collect all examples with that super class and make a balanced train/test split using a 50/50 ratio. A classifier is trained on the train split and evaluated on the test split. The performance can then be averaged over all hierarchical tasks (one for every super class) for a given dataset.

In our experiments, we finetune a pre-trained ResNet-18 network on ImageNet [41] on the hierarchical tasks. Only the output layer of the network is adjusted whilst finetuning. For the OmniPrint datasets, the network is finetuned for 20 epochs using the Adam optimizer, with a learning rate of 0.001. For the others, it trains for 10 epochs using SGD with a learning rate of 0.001. The average classification accuracy scores for every dataset are shown in Figure 4 (left). More details about the experiment, experimental settings and results are in Appendix G.

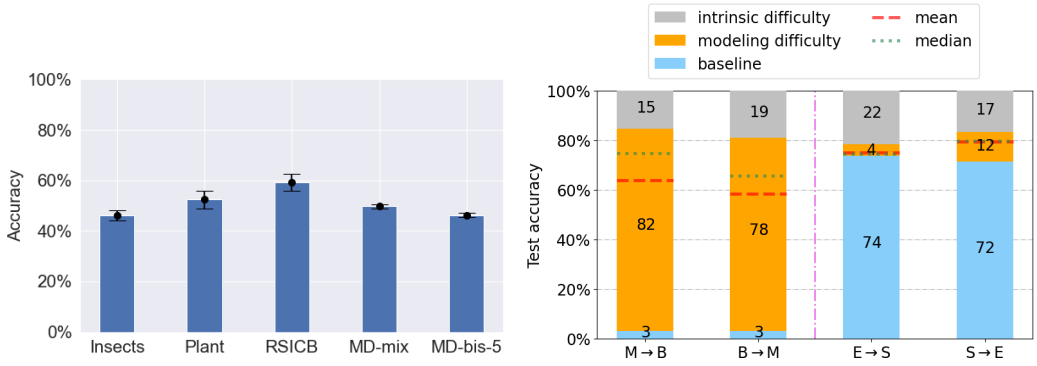


Figure 4: **Left: Hierarchical classification** accuracy for 5 datasets. We used abbreviations for the datasets. **Right: Unsupervised domain adaptation** results. $X \to Y$ means X is the source domain and Y is the target domain, $X, Y \in M, B, E, S$. M means OmniPrint-MD-mix, B means OmniPrint-MD-5-bis, E means Mini RESISC, S means Mini RSICB. Five methods are tested for each DA task, the height of the blue bar represents the performance of our worst performing method, the top of the orange bar represents the performance of our best performing method. Mean and median are computed over all methods tried.

6.4 Transfer learning: Unsupervised Domain Adaptation

Some datasets of Meta-Album can be used to benchmark domain adaptation (DA) [7]. These algorithms tested have access to both images and labels in the source domain, but only to images in the target domain. Their objective is to predict the target domain labels. We use two pairs of datasets in our experiments: The first pair is OmniPrint-MD-mix (M) and OmniPrint-MD-5-bis (B). Inspired by previous DA benchmarks [20, 49], we only used 31 randomly sampled characters (out of 706). The second pair is Mini RESISC (E) and Mini RSICB (S), which have 6 common classes. For all the four datasets, each class contains 40 examples. This yields 4 possible DA tasks, we tested each one with 5 unsupervised DA methods [56]: DAN [29, 53], DANN [11], DeepCoral [47], DAAN [59] and DSAN [63]. The experimental results are summarized in Figure 4 (right).

We observe that these two pairs are both symmetric in the sense that exchanging source domain and target domain has little influence on the methods' performance. On the other hand, while the

performance of low-end methods differs significantly for these two pairs, the high-end methods always have similar performance. More details can be found in Appendix H.

7 Discussion and conclusion

We presented **Meta-Album**, a new benchmarking framework for few-shot image classification that is both practical, since datasets are small enough to train models on regular GPUs, as well as robust since algorithms are evaluated on datasets from multiple domains. As such, it is ideal to evaluate the generalization capabilities of meta-learning and transfer learning techniques. It can also be used for hierarchical classification as well as domain adaptation, due to the presence of overlapping classes between datasets. In addition, the benchmark could be used for cross-domain few-shot learning [13, 52, 3] where an algorithm is trained on one dataset and evaluated on other datasets.

We evaluated the utility of our benchmark in several of these settings. In few-shot learning experiments, we found that the different datasets differ in difficulty, but all are amenable to learning. Fine-tuning a pre-trained model with Imagenet is quite competitive compared to meta-learning techniques for few-shot image classification. Moreover, meta-baseline authors [4] also reach the same conclusion. Meanwhile, when the target dataset is very different from the source dataset which the pre-trained model learns from (often Imagenet), meta-learning seems to work better than transfer learning. This finding echoes another publication comparing transfer learning and meta-learning [8]. Also for hierarchical classification, we find that a given algorithm obtains quite different scores on different datasets. Finally, in unsupervised domain adaptation, we also find interesting differences between methods on different pairs of datasets.

To encourage researchers to run their algorithms on our benchmark under a unified protocol and with well defined resources, all the public datasets will be made available in the Codabench platform as soon as the MetaDL challenge is over. We will offer free GPU workers. We also encourage researchers to submit new datasets to further enrich the benchmark. To ensure the quality of these and future datasets, we provide quality control software based on Factsheets and GradCAM [43]. The CodaBench platform also allows us to rerun algorithms on different extended versions of the benchmark over time and could include novel metrics in the future as well.

Future work There are several ways in which the benchmark can still be improved, and better baseline results could be established. First, several datasets do not have super-classes, hence we invite the community to contribute more datasets with such a hierarchical structure. Second, in our few-shot learning experiments, we have used the same hyperparameters for every dataset. It would be interesting to perform hyperparameter influence experiments on the baselines to measure their sensitivity to the choice of dataset. Moreover, all few-shot learning experiments in this paper were limited to the *within-distribution* setting, where learning algorithms are evaluated on tasks coming from the same dataset as the one used for training. Future work should investigate the *out-of-distribution*/cross-domain performance by measuring the ability of learning algorithms to learn tasks from different datasets than the one used for training. Lastly, it would also be interesting to perform cross-domain meta-training, meaning that training tasks may come from different datasets instead of from a single one. Investigating whether this way of training improves cross-domain performance (on unseen datasets) would be another direction for future research. In addition, the meta-training time and adaptation time could be separately recorded for each experiment.

In all, we hope that by making this benchmark publicly available, we as a community will gain new insight into the generalization abilities of few-shot classification methods. We hope this will lead to better machine learning methods that can learn from small amounts of data in many new domains.

Acknowledgments and Disclosure of Funding

We gratefully acknowledge the data owners/creators: **Insects**: Grégoire Loïs, Colin Fontaine and Jean-Francois Julien from *National Museum of Natural History Paris, France* and *SPIPOLL Science project*; **Plankton**: Heidi M. Sosik, Emily E. Peacock, Emily F. Brownlee and Eric Orenstein from *Woods Hole Oceanographic Institution, United States*; **Textures DTD**: Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed and Andrea Vedaldi, the Authors of Describable Textures Dataset (DTD); **KTH TIPS and KTH TIPS 2**: Eric Hayman, Barbara Caputo, Mario Fritz, P. Mallikarjuna and Alireza Tavakoli Targhi from *KTH Royal Institute of Technology in Stockholm*; **Kylberg Texture**: Gustaf Kylberg from *Uppsala University, Sweden*; **UIUC Textures**: Jean Ponce, Svetlana Lazebnik and Cordelia Schmid from *University of Illinois Urbana-Champaign*; **Plant Village**: Sharada Mohanty, David Hughes, and Marcel Salathé, from *EPFL Switzerland* and *Penn State University*, J. Arun Pandian and G. Geetharamani, from *Department of Mathematics, University College of Engineering, Anna University - BIT Campus and Department of Computer Science and Engineering, M.A.M. College of Engineering and Technology, Tiruchirappalli, India*; **Medicinal Leaf**: S Roopashree, J Anitha, from *Visvesvaraya Technological University, R V Institute of Management, Dayananda Sagar University, India*; **RESISC45**: Gong Cheng, Junwei Han, and Xiaoqiang Lu from *Northwestern Polytechnical University, Xi'an, China*; **RSICB128**: Haifeng Li, Xin Dou, Chao Tao, Zhixiang Hou, Jie Chen, Jian Peng, Min Deng, Ling Zhao from *Central South University, Changsha, China*;

We also received useful input from many members of the TAU team of the LISN laboratory, Wei Wei Tu from 4Paradigm Inc, China, and the MetaDL technical crew: Sebastien Treguer, Jennifer (Yuxuan) He and Benjia Zhou. We also would like to thank the hundreds of volunteers involved in the SPIPOLL citizen science program who pictured and identified insects.

This work was supported by ChaLearn, the ANR (Agence Nationale de la Recherche, National Agency for Research) under AI chair of excellence HUMANIA, grant number ANR-19-CHIA-00222 and Labex Digicosme project ANR11LABEX0045DIGICOSME operated by ANR as part of the program Investissement d'Avenir Idex Paris Saclay (ANR11IDEX000302). In addition, some experiments were performed using the compute resources from the Academic Leiden Interdisciplinary Cluster Environment (ALICE) provided by Leiden University.

References

[1] Luca Bertinetto et al. "Meta-learning with differentiable closed-form solvers". In: (2019). URL: https://openreview.net/forum?id=HyxnZh0ct7.

[2] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[3] Wei-Yu Chen et al. "A Closer Look at Few-shot Classification". In: *International Conference on Learning Representations*. ICLR'19. 2019. URL: https://openreview.net/forum?id=HkxLXnAcFQ.

[4] Yinbo Chen et al. "A new meta-baseline for few-shot learning". In: *arXiv preprint arXiv:2003.04390* (2020).

[5] Gong Cheng, Junwei Han, and Xiaoqiang Lu. "Remote Sensing Image Scene Classification: Benchmark and State of the Art". In: *Proceedings of the IEEE* 105.10 (Oct. 2017), pp. 1865–1883. ISSN: 1558-2256. DOI: 10.1109/jproc.2017.2675998. URL: http://dx.doi.org/10.1109/JPROC.2017.2675998.

[6] Mircea Cimpoi et al. *Describing Textures in the Wild*. 2013. arXiv: 1311.3618 [cs.CV].

[7] Gabriela Csurka. "Domain Adaptation for Visual Applications: A Comprehensive Survey". In: *arXiv:1702.05374 [cs]* (Mar. 2017). arXiv: 1702.05374 [cs].

[8] Vincent Dumoulin et al. "Comparing Transfer and Meta Learning Approaches on a Unified Few-Shot Classification Benchmark". In: *arXiv preprint arXiv:2104.02638* (2021).

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1126–1135. URL: http://proceedings.mlr.press/v70/finn17a.html.

[10] Mario Fritz et al. "THE KTH-TIPS database". In: 2004. URL: https://www.csc.kth.se/cvap/databases/kth-tips/index.html.

[11] Yaroslav Ganin and Victor Lempitsky. "Unsupervised Domain Adaptation by Backpropagation". In: *arXiv:1409.7495 [cs, stat]* (Feb. 2015). arXiv: 1409.7495 [cs, stat].

[12] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. "Dropblock: A regularization method for convolutional networks". In: *arXiv preprint arXiv:1810.12890* (2018).

[13] Yunhui Guo et al. *A Broader Study of Cross-Domain Few-Shot Learning*. 2020. arXiv: 1912.07200 [cs.CV].

[14] Isabelle Guyon et al. "What Size Test Set Gives Good Error Rate Estimates?" In: *IEEE Trans PAMI*. 1996, pp. 52–64.

[15] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[16] Emily F. Brownlee Heidi M. Sosik Emily E. Peacock. *Annotated Plankton Images - Data Set for Developing and Evaluating Classification Methods*. 2015. DOI: 10.1575/1912/7341. URL: https://hdl.handle.net/10.1575/1912/7341.

[17] Timothy Hospedales et al. "Meta-learning in neural networks: A survey". In: *arXiv preprint arXiv:2004.05439* (2020).

[18] David P. Hughes and Marcel Salath'e. "An open access repository of images on plant health to enable the development of mobile disease diagnostics through machine learning and crowdsourcing". In: *CoRR* abs/1511.08060 (2015). arXiv: 1511.08060. URL: http://arxiv.org/abs/1511.08060.

[19] Mike Huisman, Jan N van Rijn, and Aske Plaat. "A survey of deep meta-learning". In: *Artificial Intelligence Review* (2021). ISSN: 0269-2821. DOI: 10.1007/s10462-021-10004-4.

[20] David Hutchison et al. "Adapting Visual Category Models to New Domains". en. In: *Computer Vision – ECCV 2010*. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Vol. 6314. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 213–226. ISBN: 978-3-642-15560-4 978-3-642-15561-1. DOI: 10.1007/978-3-642-15561-1_16.

[21] Pang Wei Koh et al. "WILDS: A Benchmark of in-the-Wild Distribution Shifts". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 5637–5664. URL: https://proceedings.mlr.press/v139/koh21a.html.

[22] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[23] Gustaf Kylberg. *The Kylberg Texture Dataset v. 1.0*. External report (Blue series) 35. Centre for Image Analysis, Swedish University of Agricultural Sciences and Uppsala University, Uppsala, Sweden, Sept. 2011. URL: http://www.cb.uu.se/~gustaf/texture/.

[24] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. "Human-level concept learning through probabilistic program induction". In: *Science* 350.6266 (2015), pp. 1332–1338. ISSN: 0036-8075. DOI: 10.1126/science.aab3050. eprint: https://science.sciencemag.org/content/350/6266/1332.full.pdf. URL: https://science.sciencemag.org/content/350/6266/1332.

[25] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. "A sparse texture representation using local affine regions". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.8 (Aug. 2005), pp. 1265–1278. DOI: 10.1109/TPAMI.2005.151. URL: https://hal.inria.fr/inria-00548530.

[26] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[27] Haifeng Li et al. "RSI-CB: A Large-Scale Remote Sensing Image Classification Benchmark Using Crowdsourced Data". In: *Sensors* 20.6 (2020), p. 1594. DOI: doi.org/10.3390/s20061594.

[28] Zhengying Liu et al. "Winning solutions and post-challenge analyses of the ChaLearn AutoDL challenge 2019". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), p. 17.

[29] Mingsheng Long et al. "Learning Transferable Features with Deep Adaptation Networks". In: *arXiv:1502.02791 [cs]* (May 2015). arXiv: 1502.02791 [cs].

[30] P. Mallikarjuna et al. "THE KTH-TIPS 2 database". In: 2006. URL: https://www.csc.kth.se/cvap/databases/kth-tips/index.html.

[31] Devang K Naik and Richard J Mammone. "Meta-neural networks that learn by learning". In: *International Joint Conference on Neural Networks*. Vol. 1. IJCNN'92. IEEE. 1992, pp. 437–442.

[32] Boris N. Oreshkin, Pau Rodriguez Lopez, and Alexandre Lacoste. "TADAM: Task dependent adaptive metric for improved few-shot learning". In: (2018). Ed. by Samy Bengio et al., pp. 719–729.

[33] J. Arun Pandian and G. Geetharamani. "Data for: Identification of Plant Leaf Diseases Using a 9-layer Deep Convolutional Neural Network". In: *Mendeley Data, V1* (2019). DOI: 10.17632/tywbtsjrjv.1.

[34] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[35] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[36] Patrick Pérez, Michel Gangnet, and Andrew Blake. "Poisson Image Editing". In: *ACM Trans. Graph.* 22.3 (July 2003), pp. 313–318. ISSN: 0730-0301. DOI: 10.1145/882262.882269. URL: https://doi.org/10.1145/882262.882269.

[37] Sachin Ravi and Hugo Larochelle. "Optimization as a Model for Few-Shot Learning". In: *International Conference on Learning Representations*. ICLR'17. 2017.

[38] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. "Learning multiple visual domains with residual adapters". In: *arXiv preprint arXiv:1705.08045* (2017).

[39] Mengye Ren et al. "Meta-learning for semi-supervised few-shot classification". In: *arXiv preprint arXiv:1803.00676* (2018).

[40] S. Roopashree and J. Anitha (2020). *Medicinal Leaf Dataset*. DOI: 10.17632/nnytj2v3n5.1.

[41] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[42] Jurgen Schmidhuber. "Evolutionary Principles in Self-Referential Learning". Diploma Thesis. Technische Universität München, 1987.

[43] Ramprasaath R Selvaraju et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

[44] Hortense Serret et al. "Data quality and participant engagement in citizen science: comparing two approaches for monitoring pollinators in France and South Korea". In: *Citizen Science: Theory and Practice* 4.1 (2019), p. 22.

[45] Jake Snell, Kevin Swersky, and Richard Zemel. "Prototypical Networks for Few-shot Learning". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. NIPS'17. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/cb8da6767461f2812ae4290eac7cbc42-Paper.pdf.

[46] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[47] Baochen Sun and Kate Saenko. "Deep CORAL: Correlation Alignment for Deep Domain Adaptation". In: *arXiv:1607.01719 [cs]* (July 2016). arXiv: 1607.01719 [cs].

[48] Chen Sun et al. "Revisiting unreasonable effectiveness of data in deep learning era". In: *Proceedings of the IEEE International Conference on Computer Vision*. ICCV'17. 2017, pp. 843–852.

[49] Haozhe Sun, Wei-Wei Tu, and Isabelle M Guyon. "OmniPrint: A Configurable Printed Character Synthesizer". In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*. 2021. URL: https://openreview.net/forum?id=R07XwJPmgpl.

[50] Sebastian Thrun. "Lifelong Learning Algorithms". In: *Learning to learn*. Springer, 1998, pp. 181–209.

[51] Eleni Triantafillou et al. “Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples”. In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=rkgAGAVKPr.

[52] Hung-Yu Tseng et al. “Cross-Domain Few-Shot Classification via Learned Feature-Wise Transformation”. In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=SJl5Np4tPr.

[53] Eric Tzeng et al. “Deep Domain Confusion: Maximizing for Domain Invariance”. In: *arXiv:1412.3474 [cs]* (Dec. 2014). arXiv: 1412.3474 [cs].

[54] Oriol Vinyals et al. “Matching networks for one shot learning”. In: *Advances in Neural Information Processing Systems* 29 (2016), pp. 3630–3638.

[55] C. Wah et al. *The Caltech-UCSD Birds-200-2011 Dataset*. Tech. rep. CNS-TR-2011-001. California Institute of Technology, 2011.

[56] Jindong Wang and Wenxin Hou. *DeepDA: Deep Domain Adaptation Toolkit*. https://github.com/jindongwang/transferlearning/tree/master/code/DeepDA.

[57] Yaqing Wang et al. “Generalizing from a Few Examples: A Survey on Few-Shot Learning”. In: *ACM Comput. Surv.* 53.3 (June 2020). ISSN: 0360-0300. DOI: 10.1145/3386252. URL: https://doi.org/10.1145/3386252.

[58] Zhen Xu et al. “Codabench: Benchmarking for everyone”. In: *Submitted to NeurIPS data and benchmark track 2021*. 2021. URL: https://openreview.net/forum?id=iTzsbnFOHNl.

[59] Chaohui Yu et al. “Transfer Learning with Dynamic Adversarial Adaptation Network”. In: *arXiv:1909.08184 [cs, stat]* (Sept. 2019). arXiv: 1909.08184 [cs, stat].

[60] Tianhe Yu et al. “Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning”. In: *Proceedings of the 3rd Conference on Robotic Learning*. 2021. arXiv: 1910.10897 [cs.LG].

[61] Xiaohua Zhai et al. *A Large-scale Study of Representation Learning with the Visual Task Adaptation Benchmark*. 2020. arXiv: 1910.04867 [cs.CV].

[62] Quanshi Zhang and Song-Chun Zhu. “Visual Interpretability for Deep Learning: a Survey”. In: *CoRR* abs/1802.00614 (2018). arXiv: 1802.00614. URL: http://arxiv.org/abs/1802.00614.

[63] Yongchun Zhu et al. “Deep Subdomain Adaptation Network for Image Classification”. en. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.4 (Apr. 2021), pp. 1713–1722. ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2020.2988928.

A Data preparation

Data preparation process is done in many steps which includes:

- downloading original datasets
- cleaning the data
- cropping the images from sides to get squared images
- cropping parts of images to get squared images with required objects
- Adding background to images
- resizing squared images to 128x128 size with anti-aliasing filter

A.1 Data cleaning

All the publicly available datasets are downloaded from the sources and some are requested from their owners/creators. The data is cleaned i.e. any duplicated data, corrupt data and data which is not in proper format is removed. After cleaning the datasets are formatted in a standard format i.e. the meta-data for each dataset is kept in a csv file and the images are stored in one folder. The meta-data includes: file names, categories/classes/labels, super-categories etc.

A.2 Image uniform cropping

Insects, Mini Plant, Medleaf datasets and all Remote sensing and Manufacturing datasets come in different sizes and resolutions. Usually the object of interest i.e. insect and leaf etc is at the middle of the image. These datasets are preprocessed i.e cropped from the sides. The images with portrait orientation (i.e height > width) are cropped from top and bottom to match the width of image and the images with landscape orientation (i.e width > height) are cropped left and right to match the height of image. After this preprocessing the images are in perfect squares.

Plankton dataset from ecology domain is preporcessed in a different way. The images are in different orientations based on the size and shape of the plankton. Cropping from sides in this case is not a suitable way of preprocessing because in maximum images the plankton is spread in the orientation of the image and the images are captured in a way that the maximum area of images is covered by plankton and not the background. For images in portrait orientation, the last 3-columns of pixels on both sides are duplicated multiple times to make squared canvas image. For images in landscape orientation, the last 3-rows on both sides are duplicated multiple times. A Gaussian kernel of size 29x29 is applied on the 3 rows or columns before repeating them to make squared background image. This technique makes sure that we do not introduce artifacts to the image. The original image is than pasted on the background image at the center. The end result of this preprocessing is a squared image of plankton.

All OCR datasets are generated by the OmniPrint software[49]. The images are exactly in 128x128 size and hence no preprocessing is required for these specific datasets. The software also produces some meta-data e.g. shear, stretch, rotation, font etc. which describes the character in the image.

See Figure 8 for sample raw images from some of the datasets.

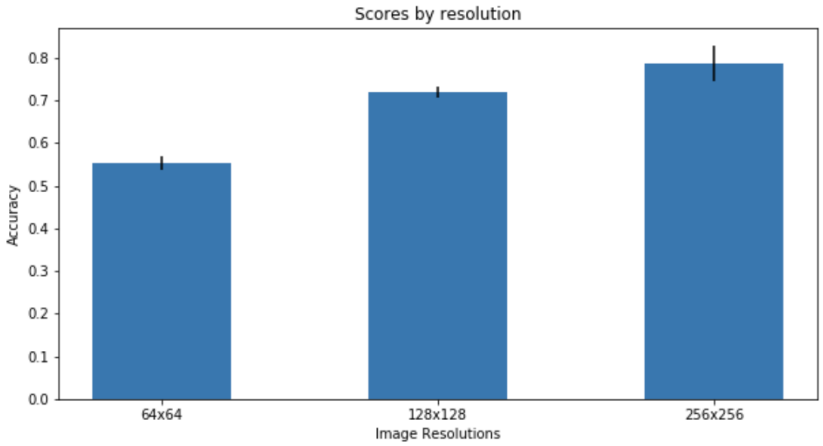


Figure 5: Image resolution comparison (64x64 vs 128x128 vs 256x256)

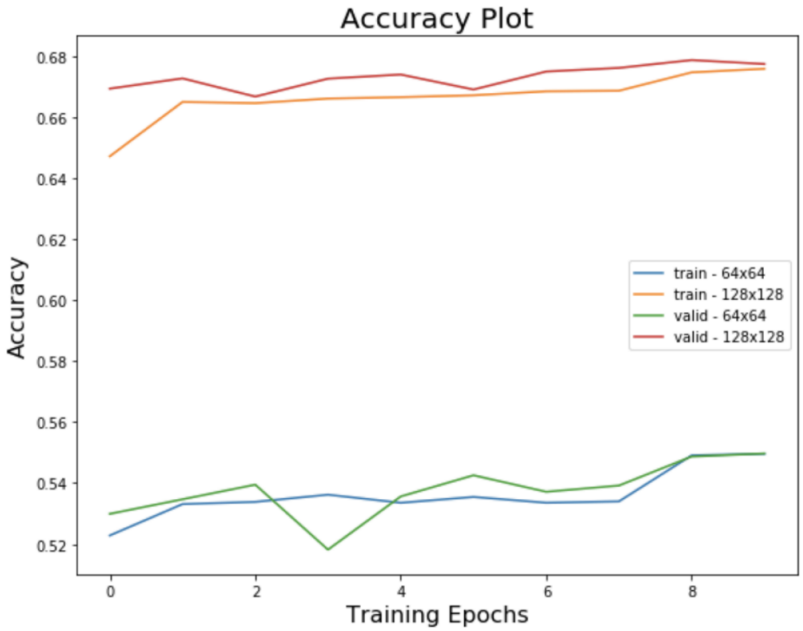


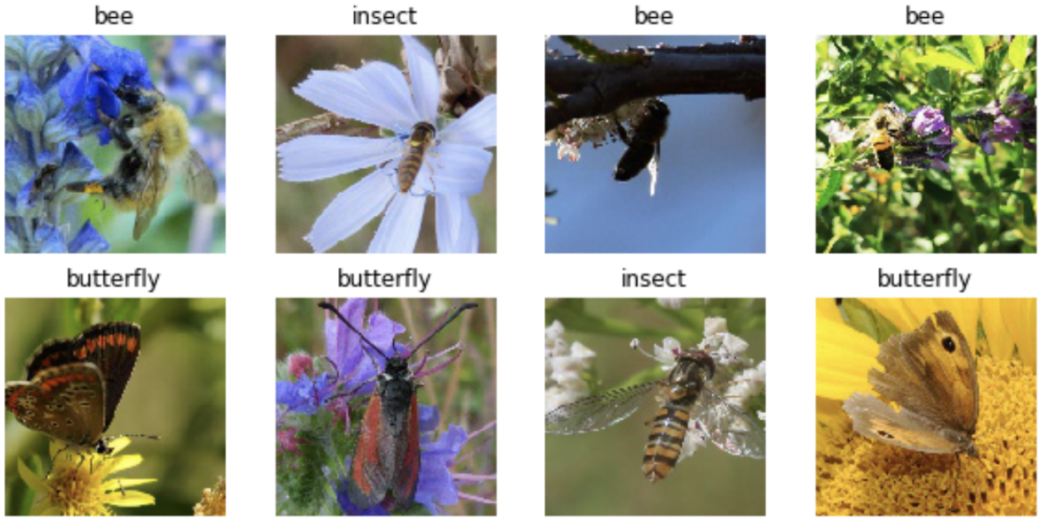
Figure 6: Image resolution comparison (64x64 vs 128x128 vs 256x256)

A.3 Image resizing

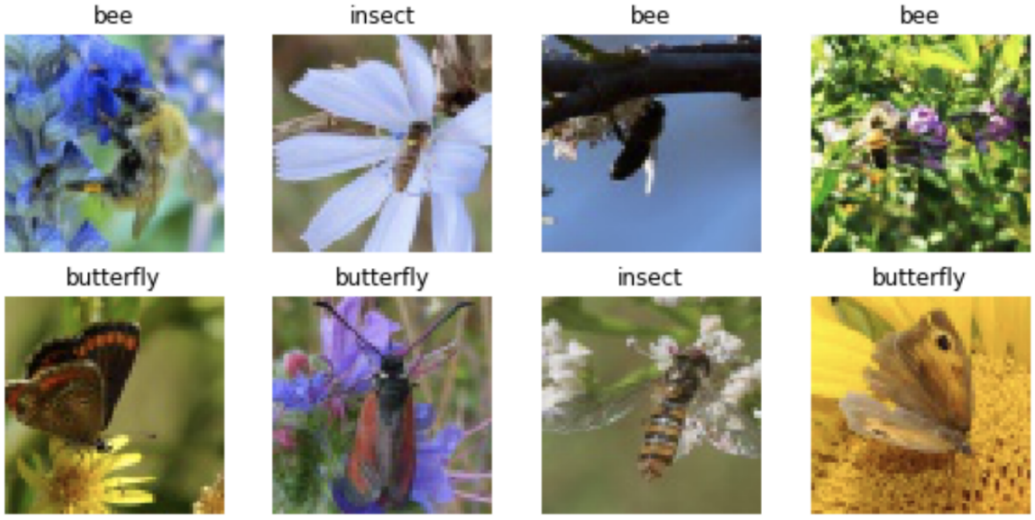
Apart from OCR datasets, all other preprocessed datasets are resized into 128x128 using open-cv with anti-aliasing filter. The anti-aliasing filter is applied to remove the aliasing effect which is normally introduced when images are down-sampled. The aliasing effect normally visible in the form of jagged edges is introduced by the loss of information during down-sampling or under-sampling.

The image resolution (128x128) is chosen based on the experiment and analysis done for different resolutions of images: 64x64, 128x128 and 256x256. For this experiment (See Figure 5), ResNet 152V2 architecture from Keras is used. It can be observed that 256x256 gives the highest scores among the observed resolutions but the dataset size also increases with a factor of 2. The images with resolution 128x128 are observed to be clearly recognizable to human eyes.

Another experiment on insects dataset for observation and comparison between 64x64 and 128x128 resolution is carried out The total images used for the experiment are 180,000 in which 135,000 are used for training while the remaining are used for validation. The accuracy plot in Figure 6 shows clearly that 128x128 produces much better results and are clearly observable too(See Figure 7). The experimental settings for this experiment includes ResNet-18 architecture, SGD optimizer, learning rate of 0.001 and 10 epochs.



(a) Sample 128x128 images



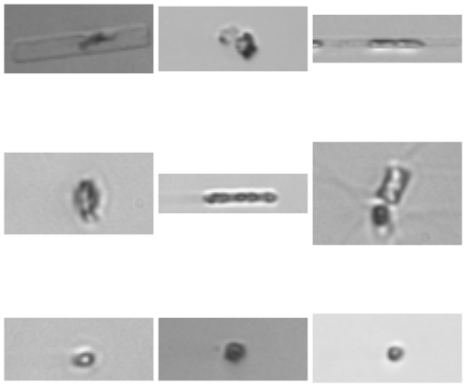
(b) Sample 64x64 images

Figure 7: Sample images in 128x128 and 64x64 resolutions

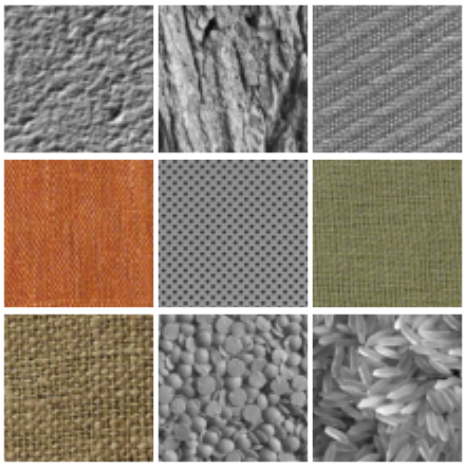
The preprocessed datasets from all domains have different number of classes and images per class. One of the goal of the Meta-Album benchmark is to prepare uniform datasets with equal number of images per class. For each dataset, the classes are picked which have at least 40 images per class. For the classes which have more than 40 images, 40 images are picked randomly for the final datasets.

The software to prepare datasets i.e add backgrounds, crop images, resize images and extract subset from a dataset is available in the preprocessing folder of the dedicated Meta-Album Github Repository[1]

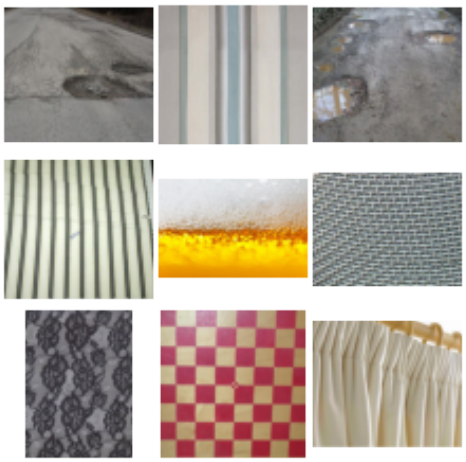
(a) Insects



(b) Plankton



(c) Textures



(d) Texture DTD

Figure 8: Meta-Album datasets raw sample images

[1] https://github.com/ihsanullah2131/meta-album/tree/master/PreProcessing

B Data quality control

B.1 Data format

In order to generate Factsheets[2] for quality control, the data is first formatted in the recommended Data format[3]. This makes easy to read and understand the data. The data format has a root directory (ideally with the dataset name and it consists of an *images* folder with all the required images for the fact-sheet generation, a *labels.csv* file with at lease two columns i.e *FILE_NAME* and *CATEGORY*(category is synonym of class/label) which shows the name of the image instance and the label of that image respectively. One more important file is *info.json* file in the root directory which has important information about the dataset and how the data should be read for factsheets. The Data format[3] repository has sample datasets for reference. To verify the data format, a python script[4] is also provided and complete details of the formatting and testing are given in the README[5] of the data format repository.

B.2 Factsheets

Factsheets[2] generation is the main experiment to examine the datasets. Preliminary report experiment can be executed either by using the python script or Ipython notebook in the Preliminary report repo. The Preliminary report generation starts with first setting some configuration parameters e.g. *CATEGORIES_TO_COMBINE*, *IMAGES_PER_CATEGORY*, *GENERATE_IMAGESHEET* and *SEED* which are used for number of classification tasks, number of shots, imagesheet for visualization of images per category and fixing a seed for regeneration of same experiment respectively. The next step is to read the data and make it ready for the training and testing with a train-test split of 0.5 i.e. for *IMAGES_PER_CATEGORY = 40* it represents a *20-shots* settings with 20 images for training and 20 images for testing. The experiment training settings include: Network architecture :ResNet-18, Loss:Cross Entropy Loss, Optimizer: SGD, Learning rate:0.001, Momentum: 0.9, LR decay period: 7, Gamma: 0.1 and Epochs: 10.

The network is trained and tested on a classification task or in meta-learning terms, on an episode. Although no meta-learning is involved in Factsheets experiments but it gives a good insight of the data in order to prepare it for mete-learning experiments i.e for the **NeurIPS MetaDL Challenge 2021**[6]. Each episode has a fixed number of categories (ways) which can be configured by the experimenter. The default configuration is 5-ways and 5-shots.

The evaluation of each task is done by looking at different metrics i.e. accuracy, loss, confusion matrix, AUC score, ROC Curves and standard error. All the metrics are visulized for each task along with sample imags (one image per category) and wrongly classified images.

All the tasks in each dataset are evaluated for each of the metrics mentioned above. In order to check if one category in a task can be differentiated significantly from other categories, AUC score from Scikit Learn [35] is used. An average AUC score of 0.5 is a random guess while a score of 0.7 or above is considered satisfactory score i.e the classes are separable/recognizable.

For each dataset the preliminary report experiment also generates a summary of all the tasks which includes the task index/name, categories in one task, average AUC score for each task and standard error for each task. Factsheets for all the datasets can be viewed in the Sample Factsheets repository[7].

The Preliminary Report generation scrip and Ipython notebook also has a configurable parameter to generate *image sheets*, a pdf file which contains all the images for each category in the dataset. The purpose of generating imagesheets is to check the images of each category with human eyes for any artifacts or bias in the images.

[2] https://github.com/ihsanullah2131/meta-album/tree/master/Factsheets

[3] https://github.com/ihsanullah2131/meta-album/tree/master/DataFormat

[4] https://github.com/ihsanullah2131/meta-album/blob/master/DataFormat/check_data_format.py

[5] https://github.com/ihsanullah2131/meta-album/blob/master/DataFormat/README.md

[6] https://autodl.lri.fr/competitions/210

[7] https://github.com/ihsanullah2131/meta-album/tree/master/Factsheets/SampleFactsheetReports

To generate a Preliminary report from the experiment for each dataset, a preliminary report python script[8] is used. This script combines the experiment results in a pdf file using the HTML template[9] to generate the preliminary report.

B.3 Additional data quality control

Additional inspection of data quality is done by checking the each output of Neural Network and analysing the value of each neuron. For a 5-class classification experiment, we visualize 5x5 plots with true labels on y-axis and NN outputs on x-axis. Each row plots shows the inputs with labels i (where $i = 0, 1, 2, 3, 4$) vs the neuron 0,1,2,3 and 4 of the output of NN for each input where number of test inputs for each class/label is 20. In addition, black bar tops shows the predicted class/label i by the NN (Figure 9). We repeat this experiment with a slight change on x-axis i.e we modify the output of NN by using $argmax$ function and we plot it instead of the output of NN (Figure 10). To check if the classes are separable we plot pairs of classes in a scatter plot and observe the separation. This plot makes it easy to remove or preprocess some classes which are too difficult to differentiate from each other(Figure 11).

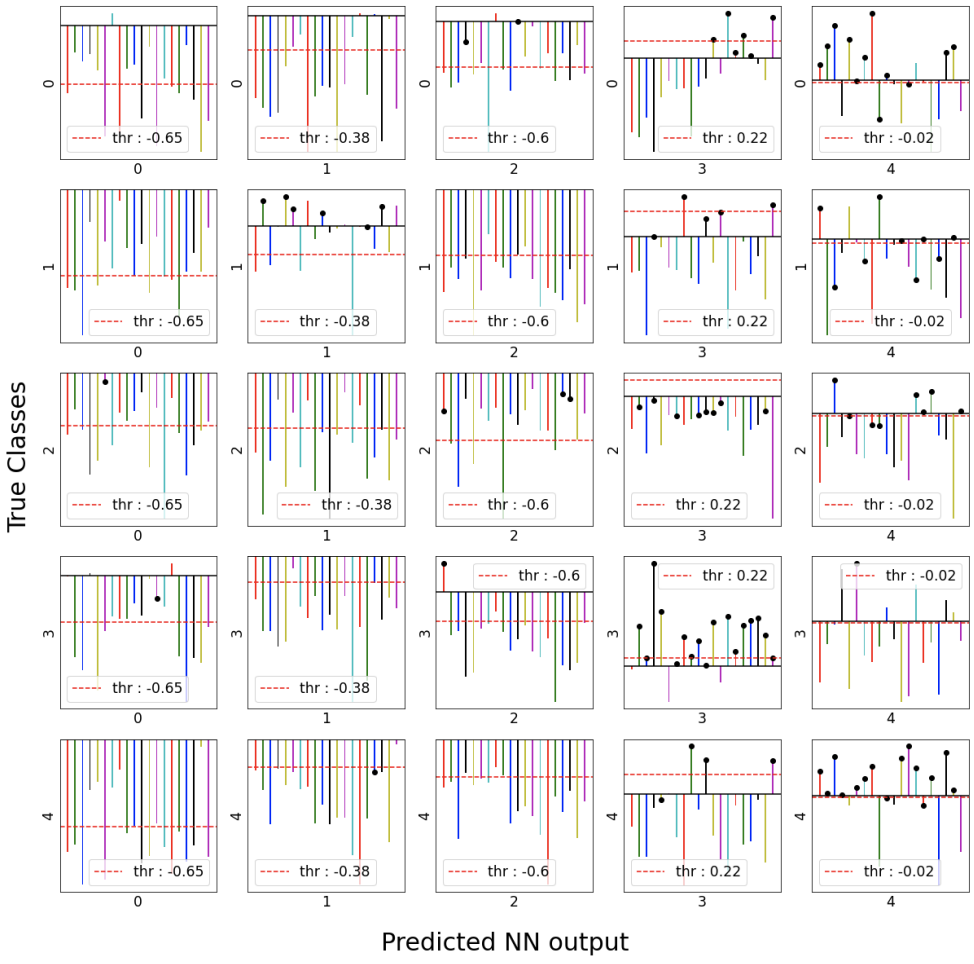


Figure 9: True classes vs output of NN

[8] https://github.com/ihsanullah2131/meta-album/blob/master/Factsheets/generate_pdf_report.py

[9] https://github.com/ihsanullah2131/meta-album/blob/master/Factsheets/template.html

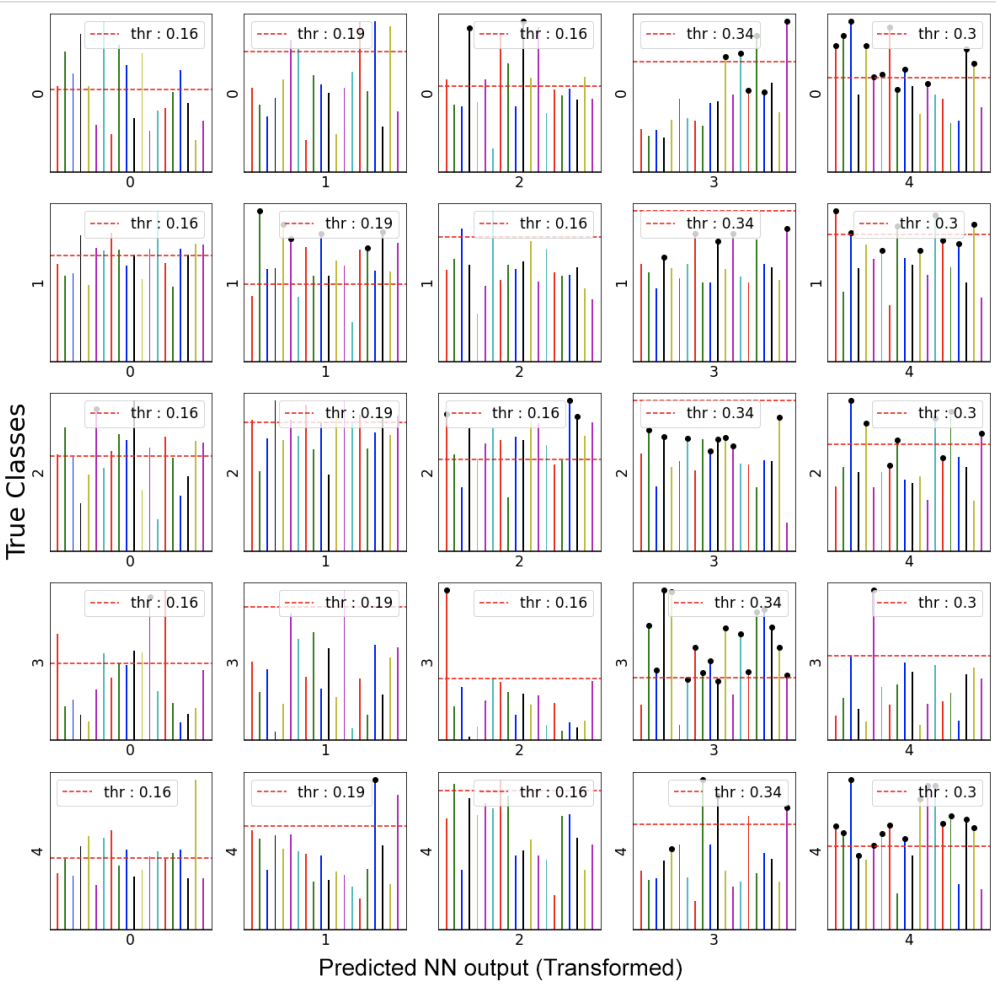


Figure 10: True classes vs transformed output of NN

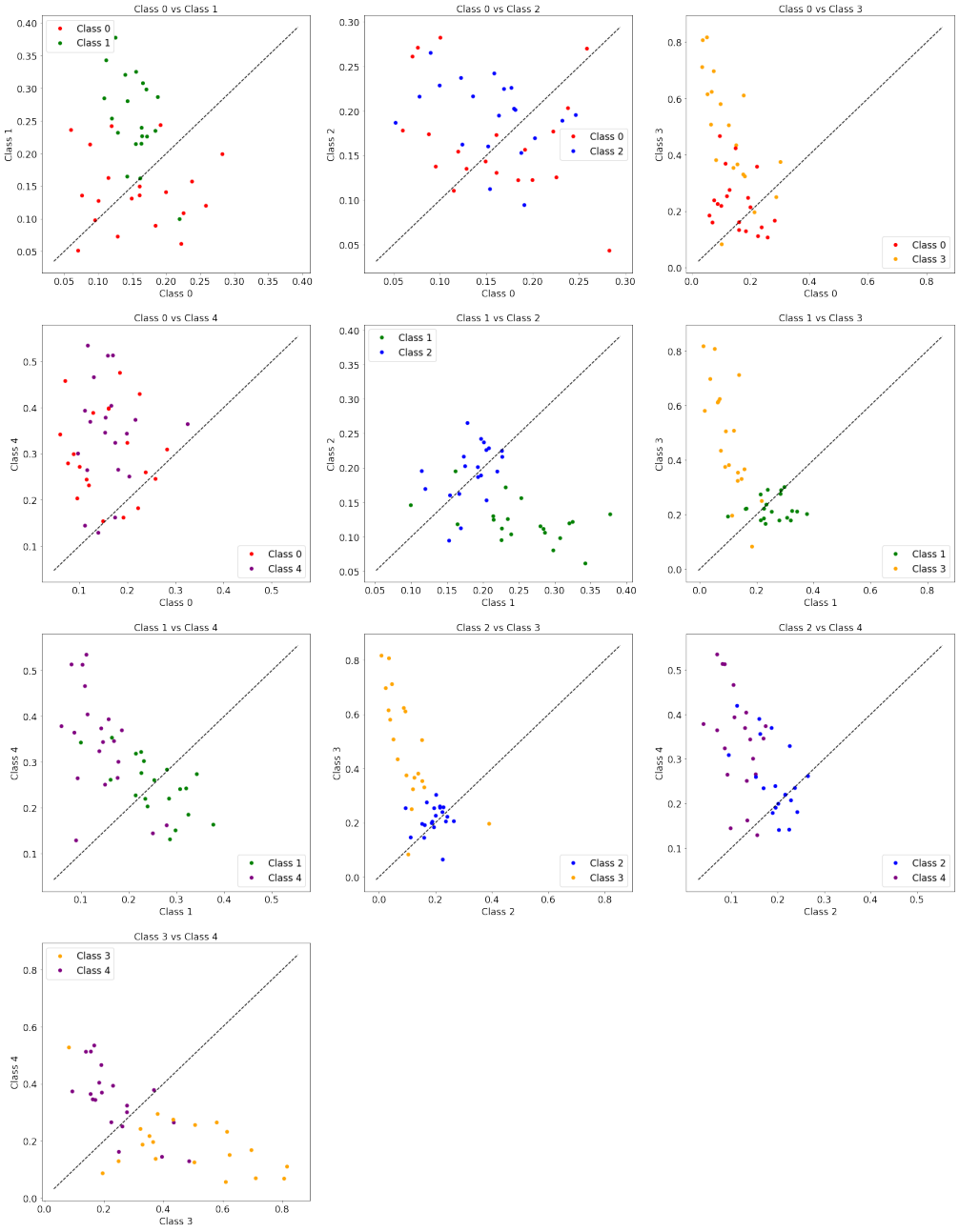


Figure 11: Pair-wise classes separation

B.4 Sample use case : Insects dataset

We take sample use case from insects dataset to explain what the preliminary report experiment generates and how do we interpret it.

We combine 5 random categories/classes together to create a classification task. Each category has 40-images per category. We use 20 images to train the last layer of ResNet-18 network and the remaining 20 images to validate the results. We visualize the train and validation accuracy and loss (See Figure 12). We also visualize the predictions by showing a confusion matrix to better understand which categories are predicted better than others (See Figure 13)

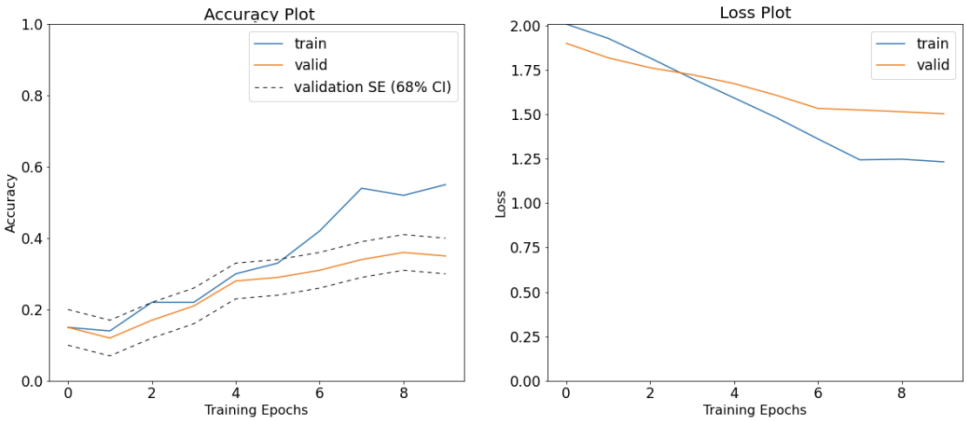


Figure 12: Insect dataset use case: Accuracy and Loss plots

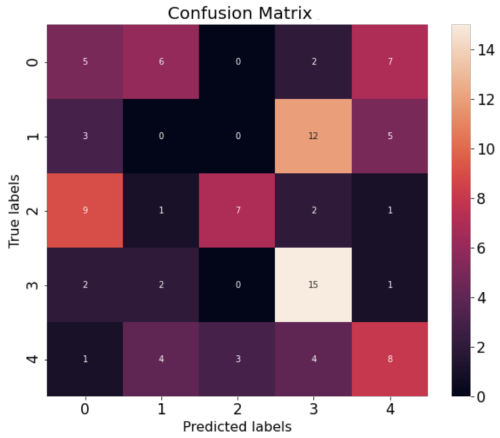


Figure 13: Insect dataset use case: Confusion matrix

We also visulaize the receiver operating characteristic (ROC) curves for each category in the experiment to analyse the performance of each class individually. For this purpose we treat each category as one vs the rest (See Figure 14). In addition we use the metric area under the receiver operating characteristic (AUROC) or AUC to analyse the class separation. We look for classes with AUC score less than 0.5 to analyse them further for any possible artifacts (See Figure 15).

The preliminary report experiment also visualizes sample images for each category (See Figure 16) and some wrongly classified images (See Figure 17).

Apart from the visualization for each episode(randomly combined 5 categories), the experiment also produces a PDF file with AUC scores per categories in decreasing order and an overall AUC histogram (See Figure 18)for clear visualization of the overall experiment.

The software to do such experiments and to generate Factsheets is available in the Meta-Album Github repository[10].

[10] https://github.com/ihsanullah2131/meta-album

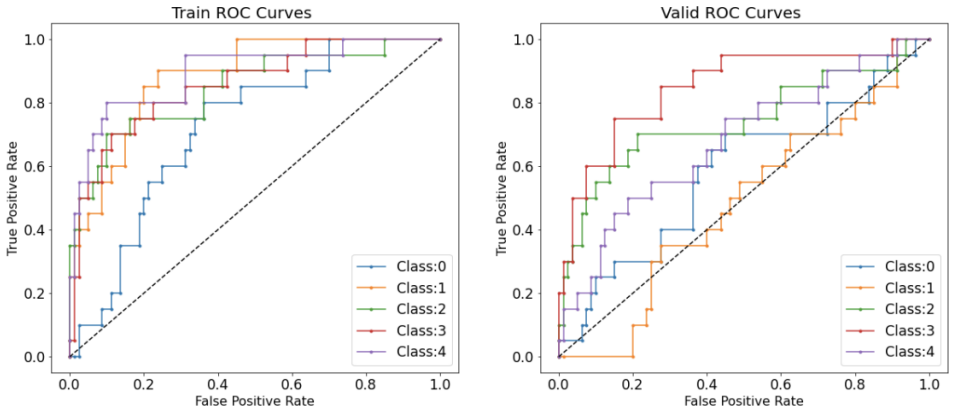


Figure 14: Insect dataset use case: ROC Curves

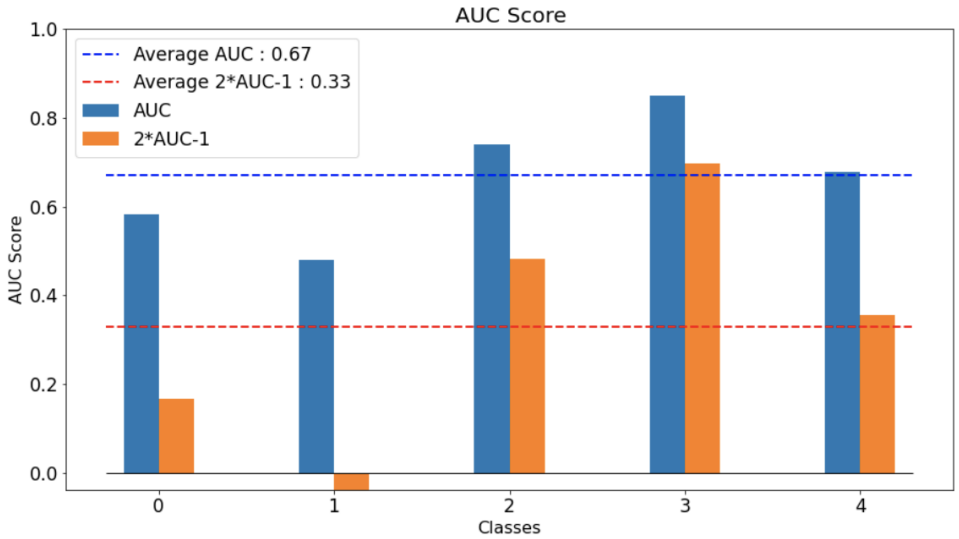


Figure 15: Insect dataset use case: AUC plot

Figure 16: Insect dataset use case: Sample images per category

Figure 17: Insect dataset use case: Wrongly classified images

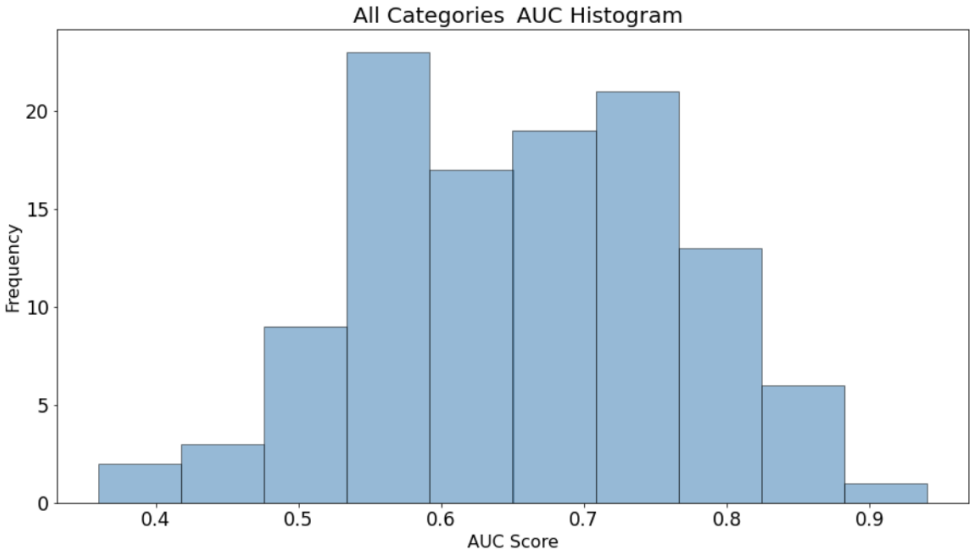


Figure 18: Insect dataset use case: Overall AUC histogram

B.5 Debugging a classification task

The preliminary experiment script and ipython notebook is created with the functionality to debug one or multiple classification tasks/episodes. It can reproduce the same results for an episode i.e. random combination of categories or combination of categories for real super-categories. We show-case the debug functionality for the Plankton dataset by debugging just one episode i.e episode index-1 of the experiment.

We have observed that repeating the same experiment again produces different results i.e. the worst performing class changes each time. From the confusion matrices (Figure 19) it can be observed that in the first confusion-matrix, category 2 is the worst performing while in the middle confusion-matrix, category 0 is the worst. Lastly in the right confusion-matrix, category 4 is the worst performing. We observed a similar inconsistency in the ROC curves (Figure 20) where the worst performing category should have lower AUC curve but that is not the case.

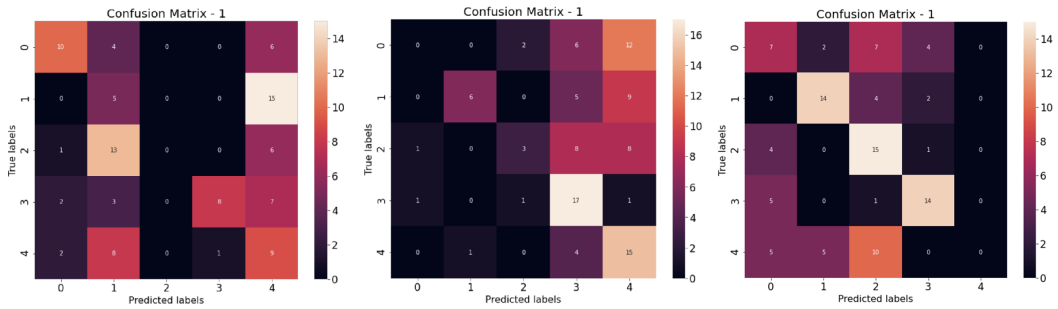


Figure 19: Debug episode-1: Confusion matrices

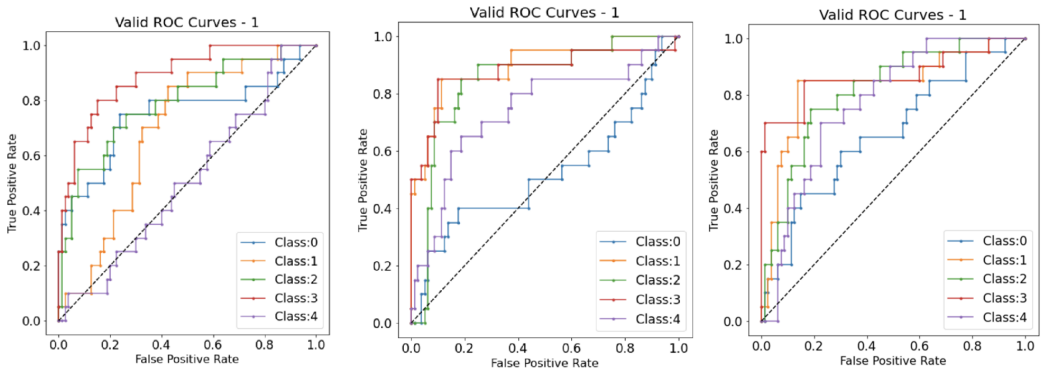


Figure 20: Debug episode-1: ROC curves

We have debug this problem a bit more by repeating the experiment with more number of images per category. We observe that with 200 images per category the results (predictions) are almost consistent for 3 independent experiments (See Figure 21). By reducing the number of images per category to 86, we observe similar results (See Figure 22). We again reduce the number of images to 40 per category and we now observe the inconsistency in the results of 3 independent experiments (See Figure 23).

We observe that the random change of worst performing category appears when the number of images are very few i.e. in our case 40 images per class means 20-images for training and 20-images for validation. By fixing the random seed of the network for training, we get exactly the same results with no more fluctuations in the worst performing category.

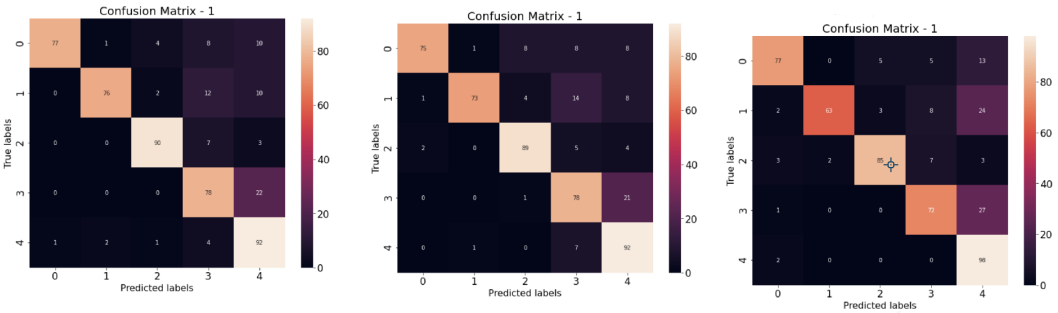


Figure 21: Debug episode-1: 200 images per category

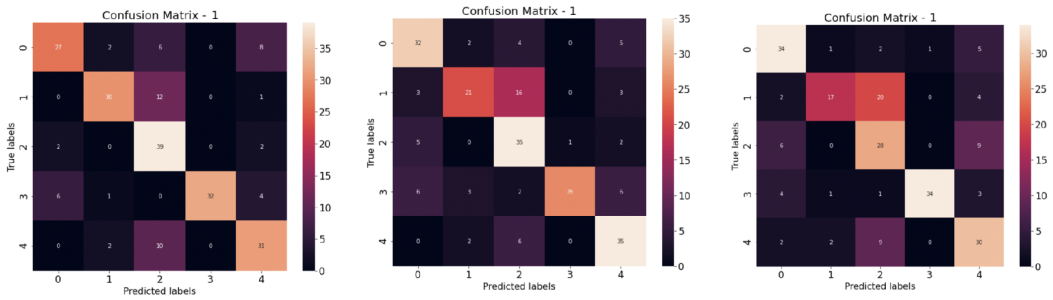


Figure 22: Debug episode-1: 86 images per category

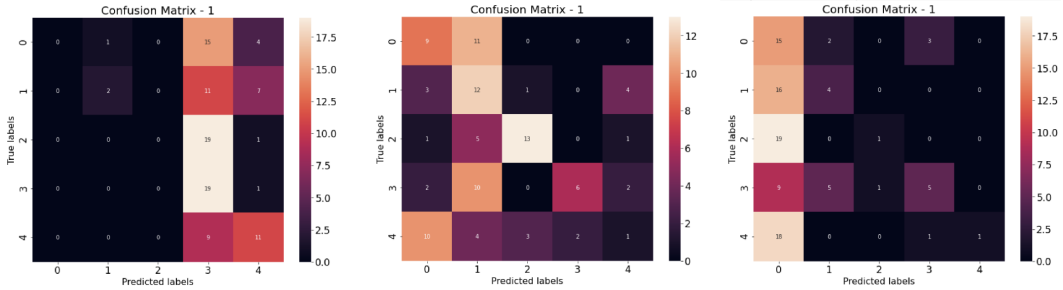


Figure 23: Debug episode-1: 40 images per category

C Datasets and meta-data

Details of Meta-Album datasets, their original sources and how the datasets are curated are explained separately in the following sections. The final Meta-Album datasets have exactly 40 images per category/class. For categories with more than 40 images per category, 40 images are selected from them randomly. The datasets come from 5 domains: ecology, bio-medicine, manufacturing, remote sensing and optical character recognition.

C.1 Ecology

Insects

The original Insects dataset [44] is createad by National Museum of Natural History, Paris[11]. It has more than 290,000 images in different sizes and orientations. The dataset has hierarchical classes which are listed from top to bottom as: Order, Super-Family, Family and Texa. Each image contains an insect in its natural environment or habitat i.e either on a flower or near to a vegetation. The images are collected by the researchers and hundreds volunteers of SPIPOLL Science project[12]. The images uploaded to a centralized server either by using the SPIPOLL website[12], Android[13] or IOS[14] mobile application.

The Meta-Album benchmark insect dataset is prepared from the original Insects dataset by carefully preprocessing the images i.e cropping the images from either sides to make squared images. These cropped images are then resized into 128x128 using open-cv with anti-aliasing filter. These preprocessed images are used in the final insects dataset which has 114 categories and total 4,560 images.

Plankton

The Plankton dataset is created by the researchers at the Woods Hole Oceanographic Institution[15]. Imaging FlowCytobot (IFCB) was used for the data collection. Complete process and mechanism is described here [16]. Each image in the dataset contains one or multiple planktons. The images are captured in a controlled environment and have different orientations based on the flow of the fluid in which the images are captured and the size and shape of the planktons.

The benchmark plankton dataset is prepared from the original WHOI Plankton's dataset. The preprocessing of the images is done by creating a background squared image by either duplicating the top and bottom most 3 rows or the left and right most 3 columns based on the orientation of the original image to match the width or height of the image respectively. A Gaussian kernal of size 29x29 is applied on the background image to blur the image. Finally the original plankton image is pasted on the background image at the center of the image. The squared background image with the original plankton image on top of it as one image is then resized into 128x128 with anti-aliasing. The final plankton dataset is extracted from these preprocessed images and it has 91 categories and total 3,640 images.

C.2 Bio-medicine

Mini Plant Village

The Plant Village dataset [16] [18, 33] contains camera photos of 17 crop leaves. The original image resolution is 256x256 px. This collection covers 26 plant diseases and 12 healthy plants. The leaves are removed from the plant and placed on gray or black background, in various lighting conditions. All labels are captured on a variety of gray background, except Corn Common rust which has black background. For the curated version, we exclude the irrelevant Background and Corn Common Rust classes from the original collection. Plant Village has 2-level label hierarchy, the supercategory is

[11] https://www.mnhn.fr/fr
[12] https://www.spipoll.org/
[13] https://play.google.com/store/apps/details?id=fr.eneo.spipoll&hl=fr
[14] https://apps.apple.com/fr/app/spipoll/id1495843067
[15] https://www.whoi.edu/
[16] https://data.mendeley.com/datasets/tywbtsjrjv/1

the crop type and the category is the disease type. We create Mini Plant Village for Meta-Album by resizing a subset from the original dataset to 128x128 px.

Medleaf

The Medicinal Leaf Dataset [17] [40] gather 30 species of healthy and mature medicinal herbs. The leaves are plucked from different plants of the same species, then placed on white uniform background. There are around 1800 images in total, captured with a mobile phone camera. The original resolution is 1600x1200 px. We create Medleaf for Meta-Album by selecting 27 classes having more than 40 images. Then we crop them at the center and resize to 128x128 px.

C.3 Remote sensing

Mini RESISC

RESISC45 [18] [5] gathers 700 RGB images of size 256x256 px for each of 45 scene categories. The data authors strives to providing a challenging dataset by increasing both within-class diversity and between-class similarity, as well as integrating many image variations. Even though RESISC45 does not propose a label hierarchy, it can be created from other common aerial image label organization scheme. We create Mini RESISC for Meta-Album by resizing a subset from the original dataset to 128x128 px.

Mini RSICB

RSICB128 [19] [27] covers 45 scene categories, assembling in total 36000 images of resolution 128x128 px. The data authors select various locations around the world, and follow China's land-use classification standard. This collections has 2-level label hierarchy with 6 supercategories: agricultural land, construction land and facilities, transportation and facilities, water and water conservancy facilities, woodland, and other lands. We create Mini RSICB for Meta-Album by resizing a subset from the original dataset to 128x128 px.

C.4 Manufacturing

Textures

The original Textures dataset is a combination of4 texture datasets: *KTH-TIPS*[20][10], *KTH-TIPS 2*[20][30], *Kylberg Textures Dataset*[21][23] and *UIUC Tectures Dataset*[25]. The data in all four datasets is collected in laboratory conditions, i.e. images were captured incontrolled environment with configurable brightness, luminosity, scale and angle. The *KTH-TIPS* dataset was collected by Mario Fritz and *KTH-TIPS 2* dataset was collected by P. Mallikarjuna and Alireza Tavakoli Targhi. Both of these datasets were prepared under the supervision of Eric Hayman and Barbara Caputo. The data for *Kylberg Textures Dataset* and *UIUC Tectures Dataset* data was collected by the original authors of these datasets. *KTH-TIPS* and *KTH-TIPS 2* datasets were created in 2004 and 2006 respectively. *Kylberg Textures Dataset* was created in September 2010 and *UIUC Tectures Dataset* in August 2005.

The Meta-Album Textures dataset is a subset of the original dataset(combination of 4 datasets). All the images are pre-processed by first cropping into perfect squared images and then resized into 128x128 with anti-aliasing filter. A subset from the original dataset is extracted with 40 images per class. The final dataset has 64 categories and total 2,560 images.

[17] https://data.mendeley.com/datasets/nnytj2v3n5/1
[18] https://gcheng-nwpu.github.io/
[19] https://github.com/lehaifeng/RSI-CB
[20] https://www.csc.kth.se/cvap/databases/kth-tips/index.html
[21] http://www.cb.uu.se/~gustaf/texture/

Texture DTD

The Textures DTD dataset is a large textures dataset[22] which consists of 5,640 images. The data is collected from Google[23] and Flicker[24] by the original authors of the paper "Describing Textures in the Wild"[6]. The data was annotated using Amazon Mechanical Turk[25]. The data collection process is mentioned on the dataset overview page[22].

For the Meta-Album benchmark, a subset of this dataset is used after preproessing it i.e. cropping the images to square images and the resizing them to 128x128 using open-cv with anti-aliasing filter. This dataset has 47 categories and total 1,880 images.

C.5 Optical character recognition

OmniPrint-MD-mix

OmniPrint-MD-mix dataset consists of 28,240 images (128x138, RGB) from 706 categories. The images are synthesized with OmniPrint [49], no further processing was done. The OmniPrint synthesis parameters are stated as follows: font size is 192, image size is 128, the strength of random perspective transformation is 0.04, left/right/top/bottom margins are all 20% of the image size, the strength of pre-rasterization elastic transformation is 0.035, random translation is activated both horizontally and vertically, rotation is within -60 and 60 degrees, horizontal shear is within -0.5 and 0.5, brightness is within 0.8333 and 1.2, contrast is within 0.8333 and 1.2, color enhancement is within 0.8333 and 1.2. The other parameters vary between images. We designed 20 settings, each setting is used to synthesize 2 images. The 20 settings are described below:

1. plain white background, random color foreground, trivial image blending (pasting)
2. plain white background, random color foreground, trivial image blending (pasting), random color outline
3. plain white background, random color foreground, trivial image blending (pasting), morphological gradient operation with elliptical kernel (kernel size is 5)
4. plain white background, image/texture foreground, trivial image blending (pasting)
5. plain white background, image/texture foreground, image/texture outline, trivial image blending (pasting)
6. random color background, trivial image blending (pasting), random color foreground
7. random color background, trivial image blending (pasting), random color foreground, random color outline
8. random color background, trivial image blending (pasting), random color foreground, morphological gradient operation with elliptical kernel (kernel size is 5)
9. random color background, trivial image blending (pasting), image/texture foreground
10. random color background augmented with random polygons, trivial image blending (pasting), random color foreground
11. image/texture background, Poisson image blending [36], random color foreground
12. image/texture background, Poisson image blending [36], random color foreground, random color outline
13. image/texture background, Poisson image blending [36], random color foreground, morphological gradient operation with elliptical kernel (kernel size is 5)
14. image/texture background, Poisson image blending [36], image/texture foreground
15. image/texture background, Poisson image blending [36], image/texture foreground, morphological gradient operation with elliptical kernel (kernel size is 5)
16. image/texture background, Poisson image blending [36], image/texture foreground, random color outline

[22] https://www.robots.ox.ac.uk/~vgg/data/dtd/index.html
[23] https://images.google.com/
[24] https://www.flickr.com/
[25] https://www.mturk.com/

17. image/texture background, Poisson image blending [36], image/texture foreground, random color outline, morphological gradient operation with elliptical kernel (kernel size is 5)
18. image/texture background, Poisson image blending [36], image/texture foreground, image/texture outline
19. image/texture background, Poisson image blending [36], image/texture foreground, image/texture outline, morphological gradient operation with elliptical kernel (kernel size is 5)
20. image/texture background, Poisson image blending [36], image/texture foreground, image/texture outline, outline size is 10

All images/textures consists of photos taken by a personal mobile phone [49].

The 706 categories are characters from:

- Armenian
  - lowercase letters
- ASCII digits
- Balinese
  - consonants
  - digits
- Basic Latin
  - lowercase letters
- Devanagari
  - digits
- Georgian
- Gujarati
  - consonants
- Hebrew
- Hiragana
- khmer
  - consonants
- Mongolian
  - basic letters
- Myanmar
  - digits
- N Ko
  - letters
  - digits
- Oriya
  - consonants
- Russian
- Sinhala
  - independent vowels
- Tamil
  - consonants
  - digits
- Telugu
  - digits

- Thai
  - consonants
- Tibetan
  - digits

OmniPrint-MD-5-bis

OmniPrint-MD-5-bis dataset consists of 28,240 images (128x138, RGB) from 706 categories. The images are synthesized with OmniPrint [49], no further processing was done. The OmniPrint synthesis parameters are stated as follows: font size is 192, image size is 128, the strength of random perspective transformation is 0.04, left/right/top/bottom margins are all 20% of the image size, the strength of pre-rasterization elastic transformation is 0.035, random translation is activated both horizontally and vertically, image blending method is Poisson Image Editing [36], rotation is within -60 and 60 degrees, horizontal shear is within -0.5 and 0.5, foreground is filled with random color, background consists of images downloaded from Pexels.

The 706 categories are characters from:

- Armenian
  - lowercase letters
- ASCII digits
- Balinese
  - consonants
  - digits
- Basic Latin
  - lowercase letters
- Devanagari
  - digits
- Georgian
- Gujarati
  - consonants
- Hebrew
- Hiragana
- khmer
  - consonants
- Mongolian
  - basic letters
- Myanmar
  - digits
- N Ko
  - letters
  - digits
- Oriya
  - consonants
- Russian
- Sinhala
  - independent vowels
- Tamil
  - consonants

  - digits
- Telugu
  - digits
- Thai
  - consonants
- Tibetan
  - digits

C.6 Meta-data

Meta-Album datasets have three sets of meta-data which comes in the files: *labels.csv*, *info.json* and *private_info.json*.

The meta-data in *labels.csv* consists of the *FILE_NAME*, *CATEGORY* and *SUPER_CATEGORY* (if any). In the case of OCR datasets, it contains some extra information about the images of characters in the data e.g. shear, stretch, rotation, font etc. *info.json* consists of meta-data (dataset name, description, number of categories and super-categories, column names etc) which is useful in reading the data, specially for data quality control. It also gives important statistics of the dataset. In *private_info.json* file, detailed meta-data is provided which consists of source, license, author, contact details with data format and statistics. This meta-data file will be kept private and will not be disclosed in the NeurIPS MetaDL Challenge 2021. The datasheets for datasets also describe some of the meta-data for the concerned datasets.

D Private Datasets (to be released within 2 years)

The Meta-Album benchmark will shortly include five additional private datasets from each domain. Such datasets will not be publicly released, initially. The motivation for keeping these datasets secret is to allow us to submit algorithms to be benchmarked to a full blind testing via code submission through the Codabench platform [58][26]. Via this platform, users' code will be tested on all 10 initial Meta-Album benchmark datasets, plus **new datasets progressively introduced**, in a fair, reproducible, and uniform environment, using docker images. To that end, we are initially providing 10 GPU compute workers in free access for the next three years (with some limitations on number of submissions per day, which will be adjusted by monitoring usage), and intend to increase such public resource progressively.

We are implementing an ever increasing benchmark, by periodically rolling in new dataset. New datasets will be initially kept private and will serve for a period of up to 2 years for:

- organizing competitions, and
- blind testing (post-challenge) submissions on the Codabench platform.

In either case, researchers can upload their algorithms, which will be executed in a sealed environment from which the datasets cannot be leaked. In this way, we will keep exposing the community to fresh data and alleviate habituation/overfitting problems.

New datasets will either be contributed by the organizing team or may be contributed by the public at large. Dataset contributors (including the authors of this paper) will have to disclose in all publications which datasets they contributed, to avoid obvious conflicts of fairness in benchmarking. We have provided detailed instructions for making contributions[27].

To bootstrap this process, we will use the 5 datasets described in this appendix. They are provided **for confidential review only**, since they will not be initially released. We commit to releasing these datasets publicly, after using them in the next challenge and after post-challenge analyses and/or blind testing benchmarks on Codabench, for a period of up to 2 years. To that end, we will either obtain re-distribution rights from the original owners (which we already secured), or provide links to the original data and our formatting code, such that users can reconstitute the dataset themselves after obtaining rights from the original owners.

[26] http://codabench.org

[27] https://github.com/ihsanullah2131/meta-album/blob/master/AddBenchmarkDataset/README.md

E License information of Meta-Album public datasets

This section provides information about the license of all the datasets of Meta-Album benchmark that we will initially publicly release.

We researched the ownership of all datasets to determine the type of usage permission. We contacted original owners by email to clarify permissions if needed, when no explicit statement of license or permission was found. Unless otherwise stated, all data are re-distributed by us under a non-commercial license. When no original license could be identified, so we use CC BY NC 4.0 by default. For commercial use, users should contact the original owners, whose contact information is found in the datasheets and in Appendix C.

Table 4: Meta-Album datasets license information

Domain	Dataset	License	Source Link an references
Ecology	Insects	CC BY NC 2.0 [a]*	SPIPOLL [44]
	Plankton	MIT [b]*	WHOI Plankton [16]
Bio-medicine	Mini Plant Village	CC BY NC 4.0[c]	Plant Village [18, 33]
	Medleaf	CC BY NC 4.0[c]	Medicinal Leaf [40]
Manufacturing	Textures	CC BY NC 4.0[c]	KTH-TIPS [10, 30] Kylberg [23] UIUC [25]
	Texture-DTD	CC BY NC 4.0[c]	DTD [6]
Remote sensing	Mini RESISC	CC BY NC 4.0[c]*	RESISC45 [5]
	Mini RSICB	CC BY NC 4.0[c]*	RSICB128 [27]
OCR	OmniPrint-MD-mix	CC BY 4.0[d]*	OmniPrint Datasets [49]
	OmniPrint-MD-5-bis	CC BY 4.0[d]*	OmniPrint Datasets [49]

[a] https://creativecommons.org/licenses/by-nc/2.0/
[b] https://github.com/hsosik/WHOI-Plankton/blob/master/LICENSE
[c] https://creativecommons.org/licenses/by-nc/4.0/
[d] https://creativecommons.org/licenses/by/4.0/

∗ The data owners/creators have confirmed the license.
For the remaining datasets, the data owners/creators have approved the redistribution of the data. We are using CC BY NC 4.0 license https://creativecommons.org/licenses/by-nc/4.0/ by default because the data owners have not chosen any particular license.

F Details for transfer learning by fine-tuning pre-trained model and configurable baseline experiments

F.1 Transfer learning by fine-tuning pre-trained model with all classes 20 shots

Experiments in this section are implemented with PyTorch [34]. We use the convolution layers from ResNet-18 pre-trained on ImageNet, then add a linear layer on top. This last linear layer has learning rate of 10^{-3}, the first convolution block has a learning rate of 10^{-7}. Convolution blocks in between these 2 extremities has a learning rate following a geometric, i.e. multiplicative progression. Learning rates are decayed by 0.1 whenever the loss stops decreasing for 2 consecutive epochs. For OmniPrint datasets, all layers have the same learning rate 10^{-3}.

The optimizer is SGD with 0 momentum, and batch size 128. Data augmentation includes random horizontal flip, random color jitter for brightness and contrast with intensity 0.1.

For OmniPrint datasets, the weight decay is 1, and number of training epochs is 20. For 2 Ecology datsets and one manufacturing one: Insects, Plankton, Hidden Ecology, the weight decay is 1, and number of training epoch is 10. For all other datasets, the weight decay is 10, and number of training epochs is 10.

Classification accuracy scores of all classes together are reported in Table 5. Here we average the validation score over 3 executions with different random seeds. Since accuracy can be considered as a binomial random variable, the 0.95 confidence interval can be estimated by $z\sqrt{\frac{p(1-p)}{n}}$ (following [14]). Here, p is the average accuracy over 3 runs with different random seeds, n is the number of validation examples for each dataset, z is the 0.975 quantile of the standard normal distribution (≈ 1.96).

Table 5: Transfer learning by fine-tuning a pre-trained model: accuracy in %, and 95% confidence interval. The 5-way 5-shot column contains the average over several tasks

Domain	Dataset	# of classes	Random guess (%)	All classes 20 shots fine-tune (%)	5-way 5-shot baseline
Ecology	Insects	114	1	39.94 ± 2.01	34.10 ± 1.08
	Plankton	91	1	60.16 ± 2.25	39.19 ± 2.40
	Hidden Ecology	102	1	85.83 ± 1.51	66.17 ± 2.61
Bio-medicine	Mini Plant Village	37	3	83.73 ± 2.66	52.60 ± 4.83
	Medleaf	27	4	92.40 ± 2.23	57.22 ± 7.82
	Hidden Bio-medicine	51	2	44.86 ± 3.05	36.40 ± 2.66
Manufacturing	Textures	64	2	94.27 ± 1.27	60.17 ± 3.92
	Texture DTD	47	2	54.18 ± 3.19	53.93 ± 3.29
	Hidden Manufacturing	250	0.4	97.95 ± 0.39	77.6 ± 2.03
Remote Sensing	Mini RESISC	45	2	70.33 ± 2.98	46.39 ± 3.95
	Mini RSICB	45	2	90.77 ± 1.89	53.25 ± 4.66
	Hidden Remote Sensing	46	2	49.96 ± 3.23	43.60 ± 4.05
OCR	OmniPrint-MD-mix	706	0.1	63.28 ± 0.80	30.89 ± 0.50
	OmniPrint-MD-5-bis	706	0.1	68.07 ± 0.77	28.83 ± 0.41
	Hidden OCR	703	0.1	61.85 ± 0.80	32.05 ± 0.52

F.2 Alleviate overfitting during fine-tuning

We start by using features extracted by a frozen convolution encoder pre-trained on ImageNet. However, using these features leads to very low performance. After all, our datasets are quite different from ImageNet. Fine-tuning the convolution encoder causes severe overfitting, as the dataset is small. Using strong weight decay helps reduce overfitting. Another simple and effective technique is to inject Dropout [46] between higher convolution layers while retraining the whole network. Dropblock [12], or spatial Dropout on contiguous area, gives almost the same benefit.

Different combination of weight decay and dropout are shown in Figure 24. Experiments are performed with a Resnet18 [15] model pre-trained on Imagenet [41]. Then, we inject dropout layer between every convolution layer in the last 2 convolution blocks, and retrain the whole network. The 2 best combinations are low dropout probability (0.1 between every convolution layer for the last 2 convolution blocks) + strong weight decay (10), or no dropout + very strong weight decay (20). These 2 settings offer the best trade-off between validation score and overfitting.

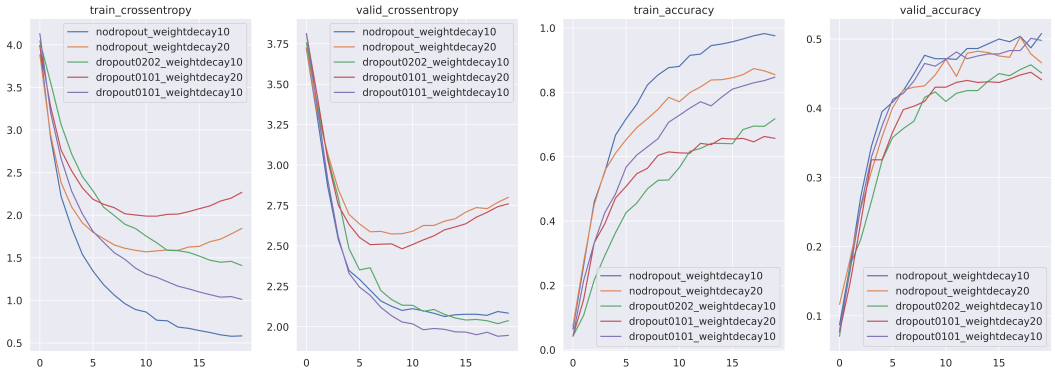


Figure 24: Effects of weight decay and dropout on the Hidden Bio-medicine dataset

F.3 5-way-5-shot configurable baseline

First, we partition all labels into distinct subgroups of 5. For example, a dataset contains 12 labels numbered from 1 to 12. By choosing the 5 ways 5 shots settings, we have 2 tasks of 5 ways classification. Suppose the 1st task contains all odd labels (1, 3, 5, 7, 9) and the second task contains all even label (2, 4, 6, 8, 10). The last 2 labels (11, 12) are discarded.

Next, we randomly sample 5 training (or support) examples for each label. The remaining 35 images of each label will be the validation (or query) examples. A model is trained on 25 examples (5 labels x 5 images), then validated on 175 examples (5 labels x 35 images). We repeat this process for all partitioned tasks.

This experiment is different from the Finetuning model of the few-shot learning experiments (Section 6.1), in which it trains on all meta-training classes together, instead of partitioning the mete-training labels into smaller subgroups to make tasks.

For this experiment, we use frozen convolution layers pre-trained on ImageNet, and update only the last linear classifier layer. The learning rate is 10^{-3}. Weight decay is set to 10, and dropout layers are inserted as described in the previous section. The remaining setup is similar to the All classes 20 shots experiments.

Average accuracies over many 5-way 5-shot classification tasks are reported in Table 5. This experiment is repeated 3 times with different random seeds for each dataset. The 0.95 confidence interval is calculated as $t\frac{s}{\sqrt{n}}$. Here, s is the sample standard deviation of validation scores for all tasks from a dataset. n is the number of 5-way 5-shot classification tasks for a dataset. t is the t-value at level 0.975 of the Student's t-distribution with $n-1$ degrees of freedom.

F.4 Hardware and compute resources

Each execution uses a single GPU among: Tesla K80, Tesla V100-PCIE-32GB, Tesla V100-SXM2-32GB. Experiment time varies from 5 - 40 minutes per dataset depending on the size.

The code for running these experiments can be found here:
https://github.com/phanav/meta-album

G Hierarchical classification

For hierarchical classification, we have chosen 5 datasets i.e. Insects from Ecology domain, Plant Village from Bio-medicine domain, Mini RSICB from Remote sensing domain, and OmniPrint-MD-mix and OmniPrint-MD-5-bis from OCR domain. The specification which qualifies them for this experiment is that these datasets have hierarchical classes i.e they have super-categories and then each super-category has a several categories. The experiments are performed on single GPUs (Tesla K80, Tesla V100-PCIE-32GB, Tesla V100-SXM2-32GB) and take at most 4 hours of wallclock time per dataset.

The experimental settings for Insects, Plant Village and Mini RSICB include: ResNet-18 architecture pre-trained on ImageNet. Only the last layer of the network is trained. The optimizer used is SGD, learning rate of 0.001 and a total of 10 epochs.

The experimental settings are different for OCR datasets because of the different nature of these datasets. A pre-trained ResNet-18 model with ImageNet is fully trained with Adam optimizer, a learning rate of 0.001. We reduce the learning rate based on the validation-loss with a factor of 0.1 and with a patience value of 10. We use 20 epochs in the training.

The details of scores and standard error are shown in the Table 6. Calculation of standard error is described in Appendix F.1

Table 6: Baseline results (hierarchical classification), accuracy in %

Dataset	super-categories	categories	Hierarchical classification 20 shots (%)
Insects	29	114	46.23 ± 2.05
Mini Plant Village	9	37	52.41 ± 3.60
Mini RSICB	7	45	59.24 ± 3.21
OmniPrint-MD-mix	23	706	49.70 ± 0.82
OmniPrint-MD-5-bis	23	706	46.26 ± 0.82

H Details for unsupervised domain adaptation experiments

This section provides the experimental details of Section 6.4 of the main paper.

The 5 unsupervised domain adaptation algorithms are DAN [29, 53], DANN [11], DeepCoral [47], DAAN [59] and DSAN [63]. The implementation is from DeepDA [56] which is under MIT License. The detailed experimental results are available in Table 7.

For each combination of task and algorithm, we run 10 epochs with 5 random seeds to get the confidence interval. The 5 random seeds were fixed in advance. The backbone neural network is ResNet50 [15]. The model is optimized using SGD with 10^{-3} as the learning rate. No hyperparameter optimization was performed. The other experimental details are provided with the code at https://github.com/SunHaozhe/transferlearning/tree/ihsan_format. The experiments were run on an internal cluster with Tesla V100-PCIE-32GB and Tesla K80. The total amount of computation time is about 51 hours.

Table 7: **Detailed unsupervised domain adaptation results.** $X \to Y$ means X is source domain and Y is target domain, $X, Y \in M, B, E, S$. M means OmniPrint-MD-mix, B means OmniPrint-MD-5-bis, E means Mini RESISC, S means Mini RSICB. The 95% confidence intervals are computed with 5 random seeds.

	$M \to B$	$B \to M$	$E \to S$	$S \to E$
DAN [29, 53]	3.3 ± 0.1	$3.3 \pm 0.$	$\mathbf{78.4 \pm 4.1}$	83.3 ± 1.7
DANN [11]	83.8 ± 1.3	79.4 ± 1.5	74.3 ± 3.2	79.8 ± 2.2
DeepCoral [47]	73.5 ± 1.0	65.8 ± 1.1	74.6 ± 2.6	79.3 ± 1.0
DAAN [59]	75.0 ± 1.3	62.7 ± 1.0	73.9 ± 2.4	71.6 ± 1.6
DSAN [63]	$\mathbf{84.8 \pm 1.2}$	$\mathbf{81.0 \pm 0.6}$	75.0 ± 4.5	$\mathbf{83.4 \pm 1.2}$
Average	64.1	58.4	75.2	79.5
Median	75.0	65.8	74.6	79.8

I Verifying baseline re-implementations

In order to verify our re-implementations of the used few-shot learning baseline techniques, we ran them using the best reported hyperparameters by the original authors on the miniImageNet benchmark [54, 37], with the exception of prototypical networks, for which we found that using the regular number of ways (N). That is, we use first-order MAML which makes $T = 5$ updates to train the initialization parameters and $T = 10$ updates at test time, uses SGD with an inner learning rate of 0.01, Adam with an outer learning rate of 0.001, meta-batch size of 4 on 1-shot classification and 2 on 5-shot classification, and gradient value clipping with an absolute threshold of 10. Prototypical networks use Adam with a learning rate of 0.001 as outer-optimizer and train in the same setting as the one during test time with a meta-batch size of 1. The same goes for matching networks. For finetuning, we use the same hyperparameters as [3]: Adam as optimizer with a learning rate of 0.001, a training batch size of 16, and 100 updates per task at test time (with the same optimizer). All techniques are trained on 80,000 tasks when $k = 1$ and on 60,000 tasks in the 5-shot setting. We validate the performance every 2,500 tasks and evaluate the best validation model at test time, where we average the results over 600 tasks. The 95% confidence interval is calculated as $z \cdot (s/n)$. Here, s is the sample standard deviation of the test scores of the 600 tasks, n the number of tasks, and $z \approx 1.96$ is the t-value at level 0.975 of the Student's t-distribution with $n - 1$ degrees of freedom. The experiments are performed on PNY GeForce RTX 2080TI GPUs with 11GB of VRAM.

We investigate the performance in the most commonly used settings: 5-way 1-shot classification and 5-way 5-shot classification. The results are displayed in Table 8. As we can see, the difference between the reported results and our results are small. We attribute differences to small differences to implementation differences (e.g., [3] used data augmentation while we did not) and randomness.

Table 8: A comparison of the meta-test accuracy scores (%) of our re-implementations vs the reported performances in the original papers on 5-way, 1- and 5-shot image classification on miniImageNet. The results are averaged over 3 runs. The 95% confidence intervals for reported results are omitted as they may have used different tasks (and in some cases are not reported by the original authors).

	1-shot		5-shot	
	Ours	Reported	Ours	Reported
Finetuning [3]	41.6±0.7	42.11	58.3±0.7	62.53
MAML [9]	45.9±0.8	48.07	59.9±0.7	63.15
Matching Network [54]	43.2±0.8	43.40	55.7±0.7	51.09
Prototypical network [45]	49.6±0.8	49.42	66.3±0.7	68.20

J Benchmarks/Datasets comparison

We compare the **Meta-Album benchmark** with other benchmarks and datasets in Table 9. Each row shows a benchmark/dataset while each column shows features. The last row shows our benchmark i.e. Met-Album.

The features are divided into quantitative and qualitative as mentioned below:

Quantitative features

- number of domains
- number of total images
- min/max number of classes per domain
- min/max number of images per class
- disk size of dataset/benchmark

Qualitative features

- multi-domain (the benchmark/dataset is multi domain)
- reasonable size(the disk size is managable on regular computers)
- uniform number of images per class
- uniform image size
- extendable (benchmark is extendable i.e. more datasets or domains could be added to it)

Table 9: Feature comparison between Meta-Album and other benchmarks/datasets

Benchmark/ Dataset	# of domains	# of images	min/max classes per domain	min/max images per class	size on disk	multi-domain	reasonable size	uniform # of images per class	uniform image size	extendable
Meta-Dataset	7	53,068,000	43/1,696	3/140,000	210 GB	✓	✗	✗	✗	✗
VTAB	3	2,244,000	2/397	40/1000	100 GB	✓	✗	✗	✗	✗
MS-COCO	1	328,000	80/80	9/10,777	44 GB	✗	✗	✗	✗	✗
Mini Imagenet	1	60,000	100/100	600/600	1 GB	✗	✗	✓	✓	✗
Omniglot	1	32,000	1623/1623	20/20	148 MB	✗	✓	✓	✓	✗
CUB-200	1	6,000	200/200	20/39	647 MB	✗	✓	✗	✗	✗
CIFAR-100	3	60,000	15/50	600/600	161 MB	✗	✓	✓	✓	✗
Meta-Album	**5**	**120,000**	**27/706**	**40/40**	**2.3 GB**	✓	✓	✓	✓	✓

For the benchmarks for which the creators haven't mentioned the domains of the datasets, it is tricky and ambiguous to divide the datasets into domains. We have divided the Meta-Dataset and CIFAR-100 in the following domains.

Meta-Dataset domains

The meta-dataset is divided into total 7 domains as listed below:

1. cu_birds (Birds, CUB-200-2011), fungi (FGVCx Fungi), vgg_flower (VGG Flower)
2. Omniglot
3. ilsvrc_2012 (ImageNet, ILSVRC) , mscoco (Common Objects in Context, COCO)
4. dtd (Describable Textures, DTD)
5. aircraft (FGVC-Aircraft)
6. quickdraw (Quick, Draw!)
7. traffic_sign (Traffic Signs, German Traffic Sign Recognition Benchmark, GTSRB)

CIFAR-100 domains

The CIFAR-100 is divided into total 3 domains as listed below:

1. people, reptiles, small mammals, aquatic mammals, fish, insects, large carnivores, large omnivores and herbivores, medium-sized mammals, non-insect invertebrates
2. trees, flowers, fruit and vegetables
3. vehicles 1, vehicles 2, large natural outdoor scenes, large man-made outdoor things, household furniture, household electrical devices, food containers

Important Links

Codabench Platform :
https://www.codabench.org/

NeurIPS MetaDL Challenge 2021 :
https://autodl.lri.fr/competitions/210

MetaDL Self Service :
https://competitions.codalab.org/competitions/31280

Main Github repo :
https://github.com/ihsanullah2131/meta-album

Factsheets repo :
https://github.com/ihsanullah2131/meta-album/tree/master/Factsheets

Factsheets Data Format :
https://github.com/ihsanullah2131/meta-album/tree/master/DataFormat

Data Format Check :
https://github.com/ihsanullah2131/meta-album/blob/master/DataFormat/check_data_format.py

Sample Factsheet Reports :
https://github.com/ihsanullah2131/meta-album/tree/master/Factsheets/SampleFactsheetReports

Factsheet Report Script :
https://github.com/ihsanullah2131/meta-album/blob/master/Factsheets/generate_pdf_report.py

Factsheet Report Template :
https://github.com/ihsanullah2131/meta-album/blob/master/Factsheets/template.html

Contact

For any query about the Meta-Album Benchmark, reach us out by email meta-album@chalearn.org