# Supplementary Material NeurIPS'22 Cross-Domain MetaDL competition: Design and baseline results

**Editors:** P. Brazdil, J. N. van Rijn, H. Gouk and F. Mohr

## Appendix A. Competition Rules

- **General Terms**: This challenge is governed by the General ChaLearn Contest Rule Terms, the CodaLab Terms and Conditions, and the specific rules set forth.
- **Announcements**: To receive announcements and be informed of any change in rules, the participants must provide a valid email.
- **Conditions of participation**: Participation requires complying with the rules of the challenge. Prize eligibility is restricted by US government export regulations, see the General ChaLearn Contest Rule Terms. The organizers, sponsors, their students, close family members (parents, sibling, spouse or children) and household members, as well as any person having had access to the truth values or to any information about the data or the challenge design giving him (or her) an unfair advantage, are excluded from participation. A disqualified person may submit one or several entries in the challenge and request to have them evaluated, provided that they notify the organizers of their conflict of interest. If a disqualified person submits an entry, this entry will not be part of the final ranking and does not qualify for prizes. The participants should be aware that ChaLearn and the organizers reserve the right to evaluate for scientific purposes any entry made in the challenge, whether or not it qualifies for prizes.
- **Dissemination**: The challenge is part of the official selection of the NeurIPS 2022 conference. There will be publication opportunities for competition reports co-authored by organizers and top-ranking participants.
- **Registration**: The participants must register to CodaLab and provide a valid email address. Teams must register only once and provide a group email, which is forwarded to all team members. Teams or solo participants registering multiple times to gain an advantage in the competition may be disqualified.
- **Anonymity**: The participants who do not present their results at the workshop can elect to remain anonymous by using a pseudonym. Their results will be published on the leaderboard under that pseudonym, and their real name will remain confidential. However, the participants must disclose their real identity to the organizers to claim any prize they might win. See our privacy policy for details.
- **Submission method**: The results must be submitted through this CodaLab competition site. The number of submissions per day and maximum total computational time are restrained and subject to change, according to the number of participants. Using multiple accounts to increase the number of submissions in NOT permitted, except that participants that want to enter the 2 leagues "free-style" and "meta-learning" are allowed to create a

second account under the name “originalID_2” , where originalID is their other account they use to make submissions (we will ask in the fact sheets which is which). In case of problem, send email to metalearningchallenge@googlegroups.com. The entries must be formatted as specified on the Instructions page.

- **Reproducibility**: The participant should make efforts to guarantee the reproducibility of their method, which in particular implies that they should use everywhere in their code the same random seed, as exemplified in the starting kit. In the Final Phase, all submissions will be run three times with various random seeds, and the worst performance will be used for final ranking. The participants will be given 2 weeks when results are released to scrutinize the evaluation procedure and the code of the winners.
- **Prizes**: The three top ranking participants in each league in the Final phase (blind testing) may qualify for prizes. The last valid submission in Feedback Phase will be automatically submitted to the Final Phase for final evaluation. The participant must fill out a fact sheet briefly describing their methods. There is no other publication requirement. The winners will be required to make their code publicly available under an OSI-approved license such as, for instance, Apache 2.0, MIT or BSD-like license, if they accept their prize, within a week of the deadline for submitting the final results. Entries exceeding the time budget will not qualify for prizes. In case of a tie, the prize will go to the participant who submitted his/her entry first. Non winners or entrants who decline their prize retain all their rights on their entries and are not obliged to publicly release their code.

**Discussion:** The rules have been designed with the criteria of *inclusiveness for all participants* and *openness of results* in mind. We aim to achieve inclusiveness for all participants by allowing them to enter anonymously and providing them cycles of computation (for the feedback and final phases) on our compute resources. This way, participants that do not have ample computing resources will not be limited by this and have a fair chance to win the challenge. We aim to achieve openness of results by requiring all participants to upload their base code and, afterward, fill in a fact sheet about the used methods. The information from the fact sheets will allow us to conduct post-challenge analyzes of the winners’ methods.

**Cheating prevention:** We will execute the submissions on our compute cluster to prevent participants from cheating, and the testing datasets will remain hidden in the CodaLab platform. Peeking at the final evaluation datasets will be impossible since those datasets are not even installed on the server during the feedback phase. Improperly using the data during the final phase will be prevented by never revealing the true test labels to the Learners but only showing them to the scoring program on the platform. Moreover, different sets of datasets (Set 0-2) are used in each phase to avoid domain-specific cheating and overfitting. We will also monitor submissions and reach out to participants with suspicious submission patterns. Finally, the winners will have to open-source their code to claim their prize. All other participants will individually scrutinize their code before they earn their prize.

## Appendix B. Starting kit and important links

The starting kit for this competition contains all the baseline methods described in this paper. Additionally, since anyone interested in meta-learning can participate, we also provide a tutorial with three difficulty levels:

- Beginner level (no prerequisites)
- Intermediate level (some knowledge of Python and meta-learning)
- Advanced level (solid knowledge of Python and meta-learning)

Each level includes information from previous levels. The idea of the tutorial is to explain to the participants all the necessary details about the competition, the data, and the submissions. To facilitate the usage of the starting kit, we distribute it in three different ways:

- Google Colab: https://colab.research.google.com/drive/1ek519iShqp27hW3xtRiIxmrqYgNNImun?usp=sharing
- GitHub Repository: https://github.com/DustinCarrion/cd-metadl
- Zip file: https://codalab.lisn.upsaclay.fr/my/datasets/download/a3476a08-a190-4455-adc2-3db175799c98

In addition to the starting kit, the important links are:

- Competition Site: https://codalab.lisn.upsaclay.fr/competitions/3627
- Forum of the competition: https://codalab.lisn.upsaclay.fr/forums/3627
- Contact email: metalearningchallenge@googlegroups.com
- Meta-learning challenge series website: https://metalearning.chalearn.org/
- Twitter account: https://twitter.com/MetaAlbum

## Appendix C. Competition Submission API

The participants must overwrite three pre-defined classes:

- **MetaLearner**: It contains the meta-algorithm logic and only the method `meta_fit( meta_train_gen, meta_valid_gen)` has to be overwritten. In general, a MetaLearner is meta-trained and returns a Learner to be meta-tested. However, it is not mandatory to meta-learn in this method; instead, participants are allowed to return a "hard-coded" learning algorithm (Learner).

- **Learner**: It encapsulates the logic to learn from a new unseen task. Several methods need to be overwritten:

  - `fit(support_set)`: Fits the Learner to a new unseen task.
  - `save(path)`: Saves the Learner in the specified path.
  - `load(path)`: Loads the Learner from the saved file(s).

  In general, a Learner is trained on the support set of a meta-test task and returns a Predictor to be tested on the unlabeled query set of that same task.

- **Predictor**: It contains the logic used by the Learner to make predictions once it is fitted. The `predict(query_set)` method must be overwritten to receive the unlabeled query set of a task, process it with the fitted Learner, and return the predicted labels.

## Appendix D. Competition Leagues

- **Free-style league:** Submit a solution obeying basic challenge rules (pre-trained models allowed).
- **Meta-learning league:** Submit a solution that meta-learns from scratch (no pre-training allowed).
- **New-in-ML league:** Be a participant with less than 10 ML publications, none of which have ever been accepted to the main track of a major conference.
- **Women league:** Special league to encourage women since they rarely enter challenges.
- **Participant of a rarely represented country:** Be a participant of a group that is not in the top 10 most represented countries of Kaggle challenge participants.

The same participant or team can compete in several leagues. For the leagues New-in-ML, Women, and Participant of a rarely represented country, in the case of teams, all team members must fulfill the specific requirements of the league to participate in it. Additionally, in these leagues, the participants are free to use either randomly initialized or pre-trained backbones.

The total pool prize is 4000 EUR, and it will be evenly distributed among the leagues as follows: 400 EUR for the $1^{st}$ place, 250 EUR for the $2^{nd}$ place, and 150 EUR for the $3^{rd}$ place. Furthermore, we will invite the winning participants to work on a post-challenge analysis collaborative paper.

## Appendix E. Baselines description and hyperparameters

This appendix provides a high-level description of each baseline method with the corresponding hyperparameters used for computing the results presented in this work.

### E.1. Train-from-scratch

This method is the simplest baseline since it does not perform any meta-training. Therefore, at meta-test time, it trains the backbone with the support set of each unseen task and then uses the trained backbone to predict the labels of the unlabeled query set. Table 1 shows the hyperparameters used by this method.

### E.2. Fine-tuning

It is a simple transfer learning method that pre-trains a backbone network with batches of data from the concatenated meta-training datasets. Then, at meta-test time, it freezes all the backbone layers except for the last one, which is fine-tuned with the support set of each unseen task. Lastly, the fine-tuned backbone is used to predict the labels of the unlabeled query set of each meta-test task. Table 2 shows the hyperparameters used by this method.

### E.3. Matching Networks

In a nutshell, Matching Networks is a metric-based method that performs the following steps:

1. Project the images of the support set into the feature space (output embeddings of the backbone).
2. Normalize the computed projections of the previous step.
3. Apply a one-hot encoding on the labels of the support set.
4. Project the images of the unlabeled query set into the feature space using the same backbone.
5. Normalize the computed projections of the previous step.
6. Compute the cosine similarity matrix between the normalized projections of the support and query sets.
7. Multiply the cosine similarity matrix and the one-hot encoding support set matrix.
8. Each image in the query set is assigned the label of the column with the highest value in the matrix obtained in the previous step.

During meta-training, the backbone is trained in an“episodic” way by maximizing the cosine similarity between images of the same class in the support and query sets. Please refer to the original paper to see the full details of this method Vinyals et al. (2016). Table 3 shows the hyperparameters used by this method.

### E.4. Prototypical Networks

In a nutshell, Prototypical Networks is a metric-based method that performs the following steps:

1. Project the images of the support set into the feature space (output embeddings of the backbone).

2. Compute the prototypes for each class. The prototypes are the mean vector of all examples of the same class.

3. Project the images of the unlabeled query set into the feature space using the same backbone.

4. Create a distance matrix by computing the Euclidean distance between the projections of the query set and each prototype.

5. Each image in the query set is assigned the label of the closest prototype.

During meta-training, the backbone is trained in an "episodic" way by minimizing the Euclidean distance between the projections of the query set images and their corresponding prototypes. Please refer to the original paper to see the full details of this method Snell et al. (2017). Table 3 shows the hyperparameters used by this method.

### E.5. FO-MAML

In a nutshell, FO-MAML is a method that relies on "episodic" training to find, during meta-training, the best possible weights for the backbone to achieve rapid adaptation when dealing with unseen tasks (meta-testing). It performs the following steps:

1. Forward the support set through the backbone and compute the loss.

2. Compute the gradients of step 1 and update the weights of the backbone.

3. Repeat steps 1 and 2 $T$ times.

4. Forward the query set through the backbone.

5. If meta-testing, compute the softmax of the output of the previous step and return the probability matrix; otherwise, continue with the next step.

6. Compute the loss using the output of step 4 and the labels of the query set.

7. Compute the gradients of step 6, but only update the weights of the backbone after processing $m$ meta-training tasks (meta batch size).

Note that in the original paper Finn et al. (2017), MAML computes second-order derivatives during the gradient calculations; nevertheless, this can be avoided by the first-order approximation of MAML (FO-MAML), where the second derivatives are omitted. Table 4 shows the hyperparameters used by this method.

### E.6. MetaDelta++

This baseline corresponds to the solution of the winners of the NeurIPS'21 competition on within-domain few-shot learning. In a nutshell, during meta-training, this method uses batch training to pre-train an ensemble of meta-learners composed of the same backbone but with a different classifier. Additionally, hand-crafted data augmentation (like rotation) is designed to help the pre-training process. Finally, the ensemble of meta-learners is "autoensembled" to stabilize the performance of the whole meta-learning system. The "autoensembled" backbone is used during meta-testing. Please refer to the original paper to see the full details of this method Chen et al. (2021).

### E.7. Hyperparameters

Table 1: Hyperparameters used by the Train-from-scratch baseline. Take into account that the backbone is a ResNet-18.

| Hyperparameter | Hyperparameter name in the code | Value |
|---|---|---|
| Optimizer | `opt_fn` | Adam |
| Learning rate | `lr` | 0.001 |
| Loss function | `criterion` | Cross-entropy |
| Number of training iterations | `T` | 100 |
| Batch size | `batch_size` | 4 |

Table 2: Hyperparameters used by the Fine-tuning baseline. Take into account that the backbone is a ResNet-18.

| Hyperparameter | Hyperparameter name in the code | Value |
|---|---|---|
| Optimizer | `opt_fn` | Adam |
| MetaLearner learning rate | `lr` | 0.001 |
| Learner learning rate | `val_lr` | 0.001 |
| Loss function | `criterion` | Cross-entropy |
| Number of training iterations for the Learner | `T` | 100 |
| Batch size for the Learner | `val_batch_size` | 4 |

Table 3: Hyperparameters used by the Matching Networks and Prototypical Networks baselines. Take into account that the backbone is a ResNet-18.

| Hyperparameter | Hyperparameter name in the code | Value |
|---|---|---|
| Optimizer | `opt_fn` | Adam |
| Learning rate | `lr` | 0.001 |
| Loss function | `criterion` | Cross-entropy |
| Meta-batch size | `meta_batch_size` | 1 |

Table 4: Hyperparameters used by the FO-MAML baseline. Take into account that the backbone is a ResNet-18.

| Hyperparameter | Hyperparameter name in the code | Value |
|---|---|---|
| Optimizer | `opt_fn` | Adam |
| MetaLearner learning rate | `lr` | 0.001 |
| Learner learning rate | `base_lr` | 0.01 |
| Loss function | `criterion` | Cross-entropy |
| Inner training iterations | `T` | 5 |
| Meta-batch size | `meta_batch_size` | 2 |

## Appendix F. Detailed results

This appendix provides the detailed results for the experiments shown in Section 4.3. Table 5 corresponds to the detailed results for Figure 1. Tables 6, 7, 8, and 9 correspond to the detailed results for Figure 2. Lastly, Table 10 corresponds to the detailed results for Figure 3.

Table 5: Detailed results for the comparison of "within-domain" and "cross-domain" few-shot learning using a randomly initialized and a pre-trained backbone. The corresponding 95% CIs are computed at task level over 3,000 tasks.

| Method | Within-Domain | | Cross-Domain | |
|---|---|---|---|---|
| | Random | Pre-trained | Random | Pre-trained |
| Train-from-scratch | 21.1 ± 0.8 | 33.3 ± 0.8 | 14.5 ± 0.7 | 26.9 ± 0.8 |
| Fine-tuning | 29.0 ± 0.9 | 39.4 ± 0.9 | 26.6 ± 0.8 | 28.8 ± 0.8 |
| Matching Networks | 35.1 ± 0.9 | 50.0 ± 0.9 | 25.7 ± 0.8 | 30.5 ± 0.8 |
| Prototypical Networks | 35.2 ± 0.9 | 51.4 ± 0.9 | 34.7 ± 0.9 | 40.6 ± 0.9 |
| FO-MAML | 24.8 ± 0.9 | 41.3 ± 0.9 | 21.1 ± 0.8 | 21.2 ± 0.7 |
| MetaDelta++ | 23.6 ± 0.8 | 75.8 ± 0.9 | 21.9 ± 0.9 | 62.3 ± 1.0 |

Table 6: Detailed results for the analysis of the influence of the number of ways on the performance of the baselines in the "cross-domain" setting using a randomly initialized backbone. The corresponding 95% CIs are computed at task level. TFS, FT, MN, PN, and MD++ stands for Train-from-scratch, Fine-tuning, Matching Networks, Prototypical Networks, and MetaDelta++, respectively.

| Ways | TFS | FT | MN | PN | FO-MAML | MD++ |
|---|---|---|---|---|---|---|
| 2 | 38.0 ± 5.0 | 49.7 ± 4.4 | 42.7 ± 4.9 | 49.8 ± 4.8 | 38.3 ± 4.7 | 35.2 ± 5.6 |
| 3 | 29.5 ± 4.5 | 41.2 ± 4.6 | 36.9 ± 4.4 | 45.3 ± 4.6 | 29.6 ± 4.4 | 29.3 ± 4.9 |
| 4 | 22.8 ± 3.5 | 39.9 ± 4.1 | 34.2 ± 3.7 | 44.0 ± 3.9 | 30.0 ± 3.9 | 28.5 ± 4.4 |
| 5 | 21.0 ± 3.3 | 35.8 ± 3.6 | 31.3 ± 3.3 | 39.4 ± 3.7 | 26.5 ± 3.5 | 26.8 ± 3.7 |
| 6 | 18.2 ± 3.2 | 31.9 ± 3.7 | 28.6 ± 3.6 | 38.6 ± 4.0 | 25.0 ± 3.5 | 24.4 ± 4.0 |
| 7 | 17.2 ± 3.1 | 32.8 ± 3.8 | 28.7 ± 3.6 | 37.6 ± 4.0 | 23.9 ± 3.5 | 24.1 ± 4.0 |
| 8 | 14.5 ± 2.4 | 27.5 ± 3.4 | 26.3 ± 3.2 | 34.3 ± 3.7 | 22.9 ± 3.2 | 21.7 ± 3.7 |
| 9 | 15.0 ± 2.8 | 29.2 ± 3.6 | 28.0 ± 3.4 | 37.0 ± 4.1 | 24.6 ± 3.5 | 24.4 ± 4.0 |
| 10 | 12.9 ± 2.5 | 24.4 ± 3.1 | 23.9 ± 2.9 | 34.2 ± 3.7 | 20.4 ± 3.1 | 20.4 ± 3.4 |
| 11 | 11.0 ± 2.3 | 23.4 ± 3.2 | 22.6 ± 3.1 | 32.0 ± 3.9 | 18.9 ± 3.0 | 20.3 ± 3.5 |
| 12 | 10.3 ± 2.1 | 21.2 ± 2.8 | 21.7 ± 2.7 | 31.1 ± 3.4 | 17.2 ± 2.5 | 17.6 ± 2.9 |
| 13 | 8.8 ± 1.8 | 19.4 ± 2.9 | 20.1 ± 2.8 | 29.1 ± 3.5 | 16.3 ± 2.8 | 17.0 ± 3.2 |
| 14 | 8.6 ± 1.8 | 20.5 ± 2.6 | 20.5 ± 2.7 | 31.2 ± 3.5 | 16.8 ± 2.6 | 17.6 ± 3.1 |
| 15 | 7.9 ± 1.8 | 16.6 ± 2.5 | 19.4 ± 2.9 | 26.6 ± 3.6 | 14.6 ± 2.6 | 16.2 ± 3.1 |
| 16 | 10.7 ± 2.2 | 21.8 ± 2.9 | 23.7 ± 3.3 | 35.0 ± 4.1 | 19.0 ± 3.0 | 21.8 ± 3.8 |
| 17 | 6.0 ± 1.4 | 16.1 ± 2.6 | 17.8 ± 2.7 | 26.7 ± 3.5 | 13.4 ± 2.4 | 15.2 ± 3.0 |
| 18 | 6.7 ± 1.5 | 16.2 ± 2.5 | 19.2 ± 2.8 | 27.5 ± 3.7 | 13.4 ± 2.5 | 16.9 ± 3.2 |
| 19 | 7.1 ± 1.7 | 16.2 ± 2.4 | 18.7 ± 2.9 | 28.7 ± 3.8 | 13.3 ± 2.5 | 17.3 ± 3.3 |
| 20 | 6.3 ± 1.3 | 17.0 ± 2.3 | 20.5 ± 2.8 | 28.5 ± 3.5 | 13.8 ± 2.2 | 18.3 ± 3.1 |

Table 7: Detailed results for the analysis of the influence of the number of ways on the performance of the baselines in the "cross-domain" setting using a pre-trained backbone. The corresponding 95% CIs are computed at task level. TFS, FT, MN, PN, and MD++ stands for Train-from-scratch, Fine-tuning, Matching Networks, Prototypical Networks, and MetaDelta++, respectively.

| Ways | TFS | FT | MN | PN | FO-MAML | MD++ |
|---|---|---|---|---|---|---|
| 2 | 48.6 ± 4.4 | 48.5 ± 4.9 | 50.1 ± 4.6 | 58.0 ± 4.3 | 39.2 ± 4.6 | 76.0 ± 4.7 |
| 3 | 41.4 ± 4.2 | 43.3 ± 4.6 | 43.7 ± 4.5 | 52.8 ± 4.4 | 32.8 ± 4.3 | 72.2 ± 4.7 |
| 4 | 39.1 ± 3.9 | 41.2 ± 4.0 | 41.5 ± 3.8 | 50.3 ± 4.0 | 32.0 ± 3.7 | 67.1 ± 4.8 |
| 5 | 36.7 ± 3.7 | 37.1 ± 3.5 | 37.2 ± 3.5 | 46.8 ± 3.6 | 25.7 ± 3.0 | 67.7 ± 4.1 |
| 6 | 33.1 ± 3.7 | 35.3 ± 3.6 | 34.8 ± 3.7 | 45.1 ± 4.0 | 25.2 ± 3.2 | 64.8 ± 4.5 |
| 7 | 32.5 ± 3.7 | 34.3 ± 3.7 | 33.9 ± 3.5 | 43.8 ± 4.0 | 25.1 ± 3.2 | 66.8 ± 4.3 |
| 8 | 29.9 ± 3.3 | 29.1 ± 3.4 | 30.0 ± 3.3 | 40.0 ± 3.9 | 21.4 ± 3.0 | 63.0 ± 4.3 |
| 9 | 30.6 ± 3.8 | 30.9 ± 3.5 | 31.7 ± 3.4 | 43.2 ± 4.2 | 22.1 ± 3.0 | 65.7 ± 4.3 |
| 10 | 26.2 ± 3.3 | 27.6 ± 3.1 | 29.5 ± 3.3 | 39.8 ± 3.9 | 20.0 ± 2.8 | 61.3 ± 4.4 |
| 11 | 22.7 ± 3.2 | 26.4 ± 3.2 | 28.0 ± 3.2 | 38.1 ± 4.0 | 18.9 ± 2.7 | 61.0 ± 4.4 |
| 12 | 22.4 ± 3.0 | 23.1 ± 2.8 | 25.2 ± 2.8 | 35.2 ± 3.6 | 17.0 ± 2.3 | 56.6 ± 4.1 |
| 13 | 20.6 ± 2.9 | 22.2 ± 2.9 | 24.4 ± 3.1 | 34.3 ± 3.8 | 16.3 ± 2.5 | 57.9 ± 4.4 |
| 14 | 19.7 ± 2.6 | 23.7 ± 2.7 | 26.1 ± 2.9 | 36.9 ± 3.7 | 17.7 ± 2.4 | 58.5 ± 3.9 |
| 15 | 17.9 ± 2.9 | 18.7 ± 2.6 | 22.6 ± 2.9 | 32.3 ± 3.8 | 14.5 ± 2.2 | 53.5 ± 4.6 |
| 16 | 22.7 ± 3.3 | 24.8 ± 2.9 | 28.4 ± 3.1 | 40.7 ± 4.3 | 18.4 ± 2.5 | 60.8 ± 4.3 |
| 17 | 15.2 ± 2.4 | 18.7 ± 2.6 | 21.1 ± 2.7 | 31.4 ± 3.6 | 13.0 ± 2.0 | 54.4 ± 4.1 |
| 18 | 16.0 ± 2.5 | 18.7 ± 2.7 | 21.6 ± 2.8 | 31.8 ± 3.9 | 13.0 ± 2.0 | 56.1 ± 4.3 |
| 19 | 15.3 ± 2.6 | 19.8 ± 2.6 | 23.6 ± 3.0 | 34.3 ± 4.0 | 14.1 ± 2.1 | 57.9 ± 4.3 |
| 20 | 15.7 ± 2.2 | 19.5 ± 2.3 | 23.2 ± 2.6 | 34.4 ± 3.7 | 13.3 ± 1.8 | 59.4 ± 3.6 |

Table 8: Detailed results for the analysis of the influence of the number of shots on the performance of the baselines in the "cross-domain" setting using a randomly initialized backbone. The corresponding 95% CIs are computed at task level. TFS, FT, MN, PN, and MD++ stands for Train-from-scratch, Fine-tuning, Matching Networks, Prototypical Networks, and MetaDelta++, respectively.

| Shots | TFS | FT | MN | PN | FO-MAML | MD++ |
|---|---|---|---|---|---|---|
| 1 | 8.1 ± 2.0 | 16.4 ± 2.9 | 14.2 ± 2.9 | 16.6 ± 3.1 | 10.8 ± 2.2 | 12.7 ± 3.4 |
| 2 | 12.5 ± 2.5 | 20.7 ± 3.2 | 18.6 ± 3.1 | 23.8 ± 3.5 | 15.6 ± 2.8 | 16.3 ± 3.4 |
| 3 | 12.8 ± 2.9 | 24.0 ± 3.5 | 19.6 ± 3.3 | 26.3 ± 3.6 | 17.1 ± 3.1 | 17.1 ± 3.6 |
| 4 | 13.0 ± 2.9 | 23.8 ± 3.4 | 22.0 ± 3.2 | 30.2 ± 3.7 | 18.0 ± 3.1 | 18.2 ± 3.6 |
| 5 | 14.2 ± 2.6 | 28.2 ± 3.3 | 24.2 ± 3.3 | 34.3 ± 3.8 | 20.9 ± 3.0 | 20.6 ± 3.4 |
| 6 | 15.0 ± 3.1 | 25.8 ± 3.6 | 25.0 ± 3.5 | 32.8 ± 4.0 | 21.1 ± 3.4 | 20.0 ± 3.8 |
| 7 | 14.1 ± 3.3 | 25.5 ± 3.8 | 24.4 ± 3.6 | 33.4 ± 4.2 | 19.7 ± 3.4 | 20.6 ± 4.0 |
| 8 | 15.8 ± 3.3 | 28.7 ± 3.9 | 26.9 ± 3.7 | 36.8 ± 4.2 | 22.5 ± 3.7 | 22.9 ± 3.9 |
| 9 | 16.4 ± 3.8 | 28.4 ± 4.3 | 26.6 ± 4.1 | 37.0 ± 4.5 | 23.3 ± 4.1 | 23.4 ± 4.5 |
| 10 | 14.7 ± 3.1 | 27.0 ± 3.6 | 27.5 ± 3.2 | 36.4 ± 3.9 | 21.2 ± 3.4 | 23.9 ± 3.7 |
| 11 | 15.9 ± 3.1 | 29.7 ± 3.5 | 29.1 ± 3.5 | 38.7 ± 3.9 | 24.4 ± 3.6 | 24.1 ± 4.0 |
| 12 | 14.4 ± 3.0 | 27.3 ± 3.7 | 26.8 ± 3.5 | 36.6 ± 3.9 | 22.2 ± 3.5 | 23.0 ± 4.0 |
| 13 | 15.3 ± 3.0 | 28.3 ± 3.8 | 28.7 ± 3.5 | 38.7 ± 4.0 | 22.9 ± 3.4 | 25.0 ± 3.9 |
| 14 | 15.7 ± 4.1 | 28.2 ± 4.3 | 28.5 ± 4.2 | 39.5 ± 4.7 | 23.5 ± 4.1 | 24.2 ± 4.6 |
| 15 | 15.2 ± 3.3 | 28.2 ± 4.0 | 28.0 ± 3.7 | 38.6 ± 4.6 | 22.3 ± 3.6 | 25.6 ± 4.2 |
| 16 | 18.3 ± 4.1 | 30.8 ± 4.4 | 30.7 ± 4.3 | 41.6 ± 4.3 | 24.5 ± 4.2 | 25.8 ± 4.8 |
| 17 | 14.1 ± 3.5 | 28.6 ± 4.0 | 27.6 ± 4.0 | 38.3 ± 4.4 | 22.7 ± 3.9 | 24.1 ± 4.4 |
| 18 | 13.6 ± 2.5 | 26.5 ± 3.1 | 28.1 ± 2.9 | 38.6 ± 3.5 | 22.8 ± 3.0 | 23.2 ± 3.3 |
| 19 | 16.5 ± 3.9 | 29.4 ± 4.2 | 31.3 ± 3.7 | 40.8 ± 4.1 | 26.2 ± 4.0 | 25.8 ± 4.3 |
| 20 | 16.6 ± 3.4 | 29.0 ± 3.9 | 29.2 ± 3.4 | 40.6 ± 4.0 | 23.3 ± 3.6 | 24.2 ± 4.1 |

Table 9: Detailed results for the analysis of the influence of the number of shots on the performance of the baselines in the "cross-domain" setting using a pre-trained backbone. The corresponding 95% CIs are computed at task level. TFS, FT, MN, PN, and MD++ stands for Train-from-scratch, Fine-tuning, Matching Networks, Prototypical Networks, and MetaDelta++, respectively.

| Shots | TFS | FT | MN | PN | FO-MAML | MD++ |
|---|---|---|---|---|---|---|
| 1 | 16.0 ± 2.8 | 16.9 ± 3.0 | 18.1 ± 3.2 | 23.7 ± 3.5 | 11.0 ± 2.3 | 48.5 ± 4.8 |
| 2 | 21.5 ± 3.2 | 22.0 ± 3.4 | 24.0 ± 3.4 | 29.8 ± 4.1 | 14.9 ± 2.4 | 50.6 ± 4.7 |
| 3 | 22.9 ± 3.2 | 24.2 ± 3.5 | 25.7 ± 3.6 | 32.2 ± 3.8 | 17.4 ± 2.8 | 55.3 ± 4.6 |
| 4 | 23.5 ± 3.4 | 25.0 ± 3.5 | 26.4 ± 3.4 | 35.4 ± 3.9 | 18.5 ± 2.7 | 58.4 ± 4.6 |
| 5 | 26.3 ± 3.1 | 29.7 ± 3.3 | 30.6 ± 3.4 | 41.1 ± 3.9 | 21.0 ± 2.7 | 62.7 ± 4.3 |
| 6 | 26.6 ± 3.4 | 27.6 ± 3.5 | 29.6 ± 3.6 | 38.7 ± 4.1 | 19.8 ± 3.1 | 59.5 ± 4.4 |
| 7 | 27.4 ± 3.7 | 28.4 ± 3.8 | 29.4 ± 3.7 | 39.5 ± 4.2 | 20.4 ± 3.2 | 62.8 ± 4.6 |
| 8 | 27.9 ± 3.8 | 30.7 ± 3.7 | 32.9 ± 3.6 | 43.2 ± 4.2 | 23.3 ± 3.2 | 60.7 ± 4.4 |
| 9 | 28.9 ± 4.2 | 32.3 ± 4.1 | 34.6 ± 4.2 | 43.6 ± 4.6 | 25.6 ± 4.1 | 63.0 ± 5.1 |
| 10 | 28.5 ± 3.7 | 29.9 ± 3.5 | 30.8 ± 3.5 | 43.5 ± 4.1 | 21.1 ± 3.0 | 65.2 ± 4.3 |
| 11 | 30.6 ± 3.7 | 32.9 ± 3.5 | 34.3 ± 3.6 | 45.3 ± 4.0 | 24.6 ± 3.4 | 66.6 ± 4.1 |
| 12 | 26.2 ± 3.6 | 30.3 ± 3.7 | 32.8 ± 3.6 | 43.3 ± 3.9 | 23.6 ± 3.4 | 63.6 ± 4.4 |
| 13 | 30.3 ± 3.8 | 30.2 ± 3.6 | 32.1 ± 3.5 | 44.4 ± 4.0 | 23.1 ± 3.4 | 67.6 ± 4.1 |
| 14 | 29.7 ± 4.6 | 31.0 ± 4.2 | 32.6 ± 4.1 | 44.9 ± 4.6 | 23.0 ± 3.6 | 65.8 ± 4.8 |
| 15 | 28.6 ± 3.9 | 30.6 ± 4.0 | 33.0 ± 3.7 | 44.1 ± 4.5 | 22.8 ± 3.4 | 64.7 ± 4.5 |
| 16 | 29.5 ± 4.4 | 34.5 ± 4.4 | 36.0 ± 4.1 | 47.1 ± 4.6 | 25.6 ± 4.0 | 69.2 ± 4.3 |
| 17 | 28.4 ± 4.3 | 30.9 ± 4.0 | 33.9 ± 4.1 | 44.4 ± 4.5 | 24.0 ± 3.8 | 65.5 ± 4.3 |
| 18 | 28.0 ± 3.5 | 29.5 ± 3.2 | 31.0 ± 3.0 | 43.4 ± 3.5 | 21.2 ± 2.6 | 66.4 ± 3.8 |
| 19 | 30.2 ± 4.3 | 31.5 ± 4.3 | 34.1 ± 3.9 | 45.2 ± 4.3 | 23.6 ± 3.7 | 67.0 ± 4.4 |
| 20 | 29.4 ± 4.1 | 31.0 ± 3.8 | 33.0 ± 3.6 | 45.4 ± 4.1 | 23.0 ± 3.3 | 66.5 ± 4.4 |

Table 10: Detailed results for the comparison of the difficulty level of Feedback phase datasets for "within-domain" and "cross-domain" few-shot learning. The results of Train-from-scratch use a randomly initialized backbone, while the results of MetaDelta++ use a pre-trained backbone. The corresponding 95% CIs are computed at task level over 300 tasks.

| Dataset | Within-Domain | | Cross-Domain | |
|---|---|---|---|---|
| | Train-from-scratch | MetaDelta++ | Train-from-scratch | MetaDelta++ |
| Dataset 1 | 9.5 ± 0.8 | 94.0 ± 0.7 | 7.9 ± 1.1 | 87.3 ± 1.1 |
| Dataset 2 | 9.4 ± 0.8 | 57.0 ± 1.6 | 7.0 ± 1.0 | 45.7 ± 1.7 |
| Dataset 3 | 16.2 ± 1.0 | 58.4 ± 1.4 | 9.4 ± 1.2 | 50.7 ± 1.7 |
| Dataset 4 | 61.6 ± 1.4 | 92.6 ± 1.0 | 50.2 ± 2.1 | 93.8 ± 0.6 |
| Dataset 5 | 6.8 ± 0.7 | 17.3 ± 0.9 | 5.2 ± 0.9 | 13.3 ± 1.0 |
| Dataset 6 | 48.9 ± 1.4 | 94.5 ± 0.7 | 31.9 ± 2.7 | 88.9 ± 1.0 |
| Dataset 7 | 34.4 ± 1.3 | 81.8 ± 0.9 | 18.4 ± 1.8 | 52.8 ± 1.7 |
| Dataset 8 | 10.3 ± 0.8 | 81.6 ± 0.8 | 6.7 ± 1.0 | 70.5 ± 1.4 |
| Dataset 9 | 13.2 ± 0.9 | 86.7 ± 1.2 | 7.9 ± 1.3 | 73.8 ± 1.5 |
| Dataset 10 | 0.5 ± 0.5 | 93.7 ± 0.6 | 0.7 ± 0.6 | 45.7 ± 3.4 |