

# Documentation Assessment Guidelines

## Reproducibility Survey Paper

In this document we present criteria guidelines for assigning values for cost for each dimension of reproducibility in AI papers. In this case ‘cost’ in general represents the possibility of documentation being provided in the method to make reproducibility more accessible. The main objective here is to assess for a paper what documentation the authors have shared or could have shared to lower the cost of reproducing their method.

Note that the statements here are **guidelines** and not **rules** as this process is very contextual. Examples are given for each dimension with possible criteria you may run into and how that can increase the cost. Note that sometimes you will have to interpret the situation of the method and how that will fit in with these principles. **We are only interested to evaluate these questions in regards to the presented method: Other presented baselines/comparison methods are not relevant.** Furthermore, we focus on the **main** experiments of a method: Extra experiments in the appendix are less important and only be should be considered **supplemental input to evaluating the main experiments.**

First, the guidelines per dimension are stated. Then, practical examples are given for two papers.

Each dimension’s cost starts off as its minimal value (1) and increases according to the guidelines given below.

Thus the range for each cost is [1,10]

## Implementation

*Given the documentation given by the authors on the method, how much effort would it be to re-implement the method?*

For the implementation we mainly focus on the code given by the authors for their method. In general we look for a link to the implementation of the authors. If this is not provided, we try to find any details that might help to re-implement the method. First check if the authors make their implementation available, if so start with point 1 otherwise with point 2.

1. If the available code is written by the original authors, this is considered part of the documentation. This URL must be stated in the review in the form of “The authors present their implementation online (\$URL\$).” The content of the repository must be checked for:
  - Details in the readme regarding installation requirements, how to execute, repository structure, data links(!)/processing. If the readme does not include clear information on how to **run** the code (Installation / examples) this increases the cost by 1.
  - Code regarding the implementation of a paper’s main method (For example: Their algorithm class or functions), data loaders and entry points. It must be skimmed for **comments** in general of how rich in information they are. Discounting comments that are cryptic: If more than 50% of core code is without comments, this increases the cost with 1.
  - If the repository structure is difficult to navigate (e.g. extremely large without index, unclear directory/file names) this increases the cost with 1.
2. If the authors **do not** provide their implementation, or parts are missing, the cost increases by 4. Then, check for other implementation details for other possible criteria to raise cost:
  - A. Regarding practical details in which language/framework/libraries their method was developed:
    - If this is not provided, should further increase the cost by 4.
    - If this is provided, but only limited (e.g. “We implemented the method in Python”), increase the cost by 3
    - If this is provided in more detail (e.g. “We implemented using Framework X, version Y, ...”), increase the cost by 2.
    - If this is provided in extensive details, with things such as design choices and other practical details, increase the cost by 1.
    - If the authors links other code repositories used for the **core** of their method implementation (e.g. “We used the SciPy implementation of Dijkstra’s Algorithm”), no additional cost is added.

B. Pseudo code definitions and general figures regarding architecture of the method in the paper (If applicable):

- If none are provided, increase the cost by 2.
- If either is provided with substantial detail, increase by 1. If both, increase by 0.

C. Any other possible practical information that can be found regarding the implementation that do not apply to any of the above categories can **decrease** the cost by up to 2 points.

## Data

*Given the data description in the documentation, how much effort would it take to either: Find the same dataset the authors used, or similar datasets and defend the comparability, or acquire one from scratch?*

Check what datasets are being used by the authors. Note that usually multiple datasets are used: In general this cost is mostly impacted by the ‘most accessible data set’ and the ‘most used dataset’. At the top it should be stated how many datasets (n) are used and how many of those are public (p) as “(p/n)” including the brackets on a separate line. If it is not clearly stated that the data is public (Either in the text, implementation or the references), assume its private. Environments/Simulators and synthetic data generators are also considered **data sets**. In case the method uses multiple data sets, the cost increase should be **averaged**, taking into account how much it is used. If for example paper X uses 3 different data sets D1, D2 and D3, but the presented results are 50% D1, and 25% D2, 25% D3, the cost should be:  $(C1 * 0.5 + C2 * 0.25 + C3 * 0.25) / 3$ . Its fine to ballpark this calculation, just write down your reasoning (“The majority of experiments”, “the private data set has a public alternative presented, therefore...”). First check below if you are dealing with datasets (First paragraph) or synthetic generators / simulators (Second paragraph).

For each dataset that is used, check that it has a description, citation, statistics and a direct link to where it can be found. If you recognise the dataset, feel free to use this information regarding public/private data.

- If there is no direct link present to where the data can be found, increase the cost by 1.

- If there is no direct link **and** no citation, increase by 2.
- If there is no description on the dataset, increase by 1.
- If there are no or limited statistics, increase the cost by 1.
- If the data set is (partially) private, increase the cost by 1-2. If the collection strategy poorly is stated, increase the cost again, by up to another 3.
- If all of the above were true, (e.g. the data set is only **named**) increase the cost by up to an **additional 5** (based on the context).

In case of simulated environments or synthetic data **generators**, check if the environments are publicly available and linked/cited, and the configurations under which they were used.

- If the code on this is not provided, either by link/implementation/citation, increase the cost by 5.
- If the process of the generator or the environment/task of the simulator is not (clearly) described, increase the cost by 2.
- If the simulation/environment has parameters and are not specified, increase by 1.
- If the synthetic generation parameters are not stated **clearly** increase the cost by up to 3.

## Configuration

*Given the (hyper)parameters, including semantic parameters, of the method: How much effort would it take to acquire the algorithm configurations used for their results, and compare against their budgetary constraints?*

Here we are interested in if the authors state what algorithm configurations or hyper parameter values they used for their experiments. Without these values, it can be costly to determine these values to reproduce the method.

1. Regarding the (hyper)parameters, check if the authors summarise their hyperparameters clearly, for example by providing a table, pseudo code or the implementation documents this:

- If this is missing, increase the cost by 3.
  - If this informally stated in the text, increase the cost by 2.
  - If this is detailed in the text, but an overview is missing, increase by 1.
2. Regarding the (hyper)parameter values:
- If the values are (almost) not specified, increase by 4.
  - If the authors do not provide all values for each experiment, increase the cost by 1.
  - If large parts of the parameter values per experiment are ambiguous, increase by 2.
3. Regarding the (hyper) parameter acquisition:
- If the authors do not state a strategy (e.g. ‘empirically’ / ‘we chose’ / ‘based on the values of bla bla et al.’ / ‘grid search’ / ‘SMAC3’), increase the cost by 1.
  - If it is not clarified under what budget the acquisition was done, increase the cost by 1.

## Experimental Setup

*Given the experimental set-up of the work, how difficult is it to set up a new experiment, similar to those presented in the original work, with the same procedure?*

Here we are interested in whether the authors clearly document under which set up the empirical evaluation was done, s.t. we can set up a new experiment under the same conditions. If there are multiple experiments, the result is the (possibly weighted if that is applicable) average.

1. The metrics are being used to evaluate the method:
- If the authors do not state these/provide citations on them, unless they are very standard in AI (‘accuracy’ or ‘F1-score’ needs no explanation) increase the cost by 1 or 2 depending on how many metrics this applies to.
2. The data that is being evaluated on:

- If there is no clear specification of its training/validation/testing data, or its evaluated on the full data set, increase the score by 1.
  - If it is not clear how the dataset is split to train / test on, increase the score by 2.
3. The strategy that is used to acquire the evaluations (Can overlap with 2):
- Is the strategy not clearly stated (E.g. multiple folds, k-fold-cross validation, single/multiple runs over seeds,..) but is implied increase by 1, if not specified at all increase by 2.
  - If strategy details are missing such as parameters/repetition (E.g. k-fold but no k given), increase by 1.
4. The aggregation of the results:
- Is it clear how the results are aggregated? If not increase by 1.
  - If there is a measure of distribution presented over the results, is it clear what type this is (Std. dev, variance, 95% CI,..)? If not increase by 1.

## Expertise

*How much effort would it take to acquire the expertise required to reproduce the work independently relying on the available documentation?*

Here we want to evaluate from a scale from 1-10 (1 being accessible and 10 requiring a lot of expertise). For this question it is the target to assess how much knowledge you have to bring or how much extra reading you have to do **outside** of the documentation given by the authors. General points to consider: How many different fields within AI are being touched upon? How well is the problem introduced, or are we expected to already understand the problem? How much mathematics / logic / proofs are presented, and how well is each introduced? The more we are expected to have previous knowledge before reading the paper, and the more complex the techniques, the higher this cost as we have to rely increasingly on previous experience or external documentation to reproduce the work. This is not by definition a bad thing, rather it means 'not everybody' will be able to reproduce the work 'out of the box'.

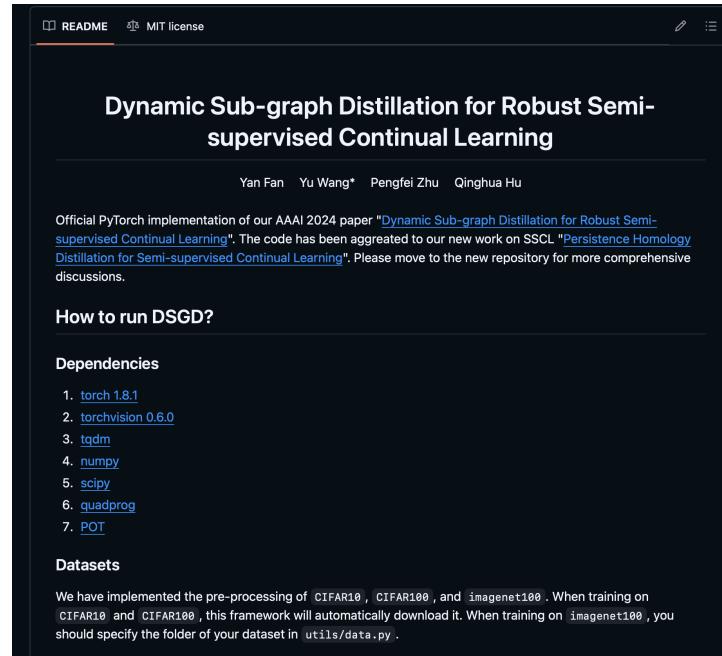
# Examples

## Dynamic Sub-graph Distillation for Robust Semi-supervised Continual Learning

AAAI 2024, [link here](#)

### Implementation [4]

The authors provide a GitHub link with their implementation. The readme is a bit short but has a lot of structured information.



Dynamic Sub-graph Distillation for Robust Semi-supervised Continual Learning

Yan Fan Yu Wang\* Pengfei Zhu Qinghua Hu

Official PyTorch implementation of our AAAI 2024 paper "[Dynamic Sub-graph Distillation for Robust Semi-supervised Continual Learning](#)". The code has been aggregated to our new work on SSCL "[Persistence Homology Distillation for Semi-supervised Continual Learning](#)". Please move to the new repository for more comprehensive discussions.

#### How to run DSGD?

#### Dependencies

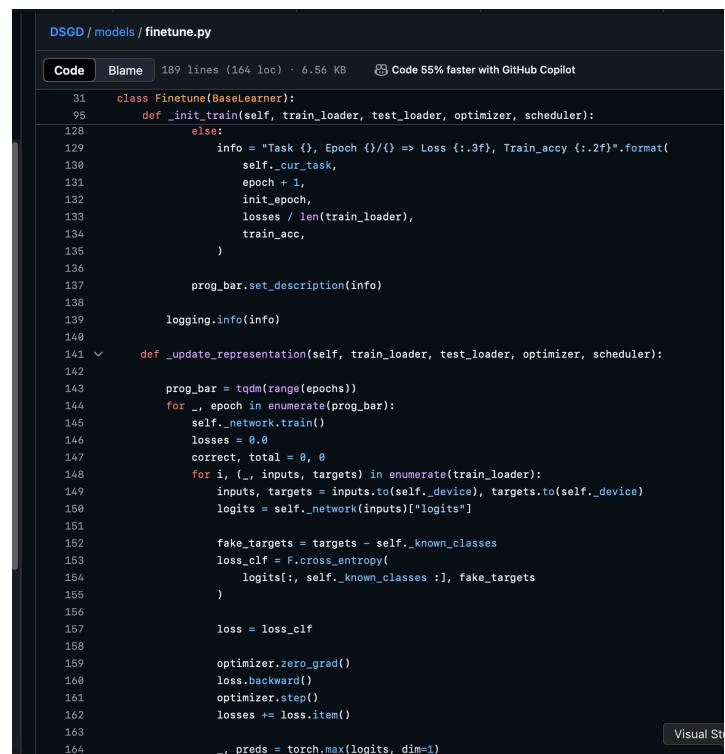
1. [torch 1.8.1](#)
2. [torchvision 0.6.0](#)
3. [tadm](#)
4. [numpy](#)
5. [scipy](#)
6. [quadprog](#)
7. [POT](#)

#### Datasets

We have implemented the pre-processing of `CIFAR10`, `CIFAR100`, and `Imagenet100`. When training on `CIFAR10` and `CIFAR100`, this framework will automatically download it. When training on `Imagenet100`, you should specify the folder of your dataset in `utils/data.py`.

However the directory structure is huge and most of the code has no comments see for example: <https://github.com/fanyan0411/DSGD/blob/main/models/finetune.py>

This means understanding the code is costly, therefore its evaluated with a **4**.



```
DSGD / models / finetune.py
Code Blame 189 lines (164 loc) · 6.56 KB Code 55% faster with GitHub Copilot
31  class Finetune(BaseLearner):
32      def __init__(self, train_loader, test_loader, optimizer, scheduler):
33          super().__init__(train_loader, test_loader, optimizer, scheduler)
34          self._network = self._network.to(self._device)
35          self._cur_task = 0
36          self._known_classes = set()
37
38          self._init_train(self, train_loader, test_loader, optimizer, scheduler)
39
40      def _init_train(self, train_loader, test_loader, optimizer, scheduler):
41          self._train_loader = train_loader
42          self._test_loader = test_loader
43          self._optimizer = optimizer
44          self._scheduler = scheduler
45
46          self._train_accy = 0.0
47          self._test_accy = 0.0
48
49          self._train_loss = 0.0
50          self._test_loss = 0.0
51
52          self._train_bar = tqdm(self._train_loader)
53          self._test_bar = tqdm(self._test_loader)
54
55          self._train_bar.set_description("Training")
56          self._test_bar.set_description("Testing")
57
58          self._train_bar.refresh()
59          self._test_bar.refresh()
60
61          self._train_bar.update(0)
62          self._test_bar.update(0)
63
64          self._train_bar.set_postfix(
65              train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
66          )
67
68          self._test_bar.set_postfix(
69              train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
70          )
71
72          self._train_bar.refresh()
73          self._test_bar.refresh()
74
75          self._train_bar.update(0)
76          self._test_bar.update(0)
77
78          self._train_bar.set_postfix(
79              train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
80          )
81
82          self._test_bar.set_postfix(
83              train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
84          )
85
86          self._train_bar.refresh()
87          self._test_bar.refresh()
88
89          self._train_bar.update(0)
90          self._test_bar.update(0)
91
92          self._train_bar.set_postfix(
93              train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
94          )
95
96          self._test_bar.set_postfix(
97              train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
98          )
99
100         self._train_bar.refresh()
101         self._test_bar.refresh()
102
103         self._train_bar.update(0)
104         self._test_bar.update(0)
105
106         self._train_bar.set_postfix(
107             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
108         )
109
110         self._test_bar.set_postfix(
111             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
112         )
113
114         self._train_bar.refresh()
115         self._test_bar.refresh()
116
117         self._train_bar.update(0)
118         self._test_bar.update(0)
119
120         self._train_bar.set_postfix(
121             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
122         )
123
124         self._test_bar.set_postfix(
125             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
126         )
127
128         self._train_bar.refresh()
129         self._test_bar.refresh()
130
131         self._train_bar.update(0)
132         self._test_bar.update(0)
133
134         self._train_bar.set_postfix(
135             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
136         )
137
138         self._test_bar.set_postfix(
139             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
140         )
141
142         self._train_bar.refresh()
143         self._test_bar.refresh()
144
145         self._train_bar.update(0)
146         self._test_bar.update(0)
147
148         self._train_bar.set_postfix(
149             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
150         )
151
152         self._test_bar.set_postfix(
153             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
154         )
155
156         self._train_bar.refresh()
157         self._test_bar.refresh()
158
159         self._train_bar.update(0)
160         self._test_bar.update(0)
161
162         self._train_bar.set_postfix(
163             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
164         )
165
166         self._test_bar.set_postfix(
167             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
168         )
169
170         self._train_bar.refresh()
171         self._test_bar.refresh()
172
173         self._train_bar.update(0)
174         self._test_bar.update(0)
175
176         self._train_bar.set_postfix(
177             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
178         )
179
180         self._test_bar.set_postfix(
181             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
182         )
183
184         self._train_bar.refresh()
185         self._test_bar.refresh()
186
187         self._train_bar.update(0)
188         self._test_bar.update(0)
189
190         self._train_bar.set_postfix(
191             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
192         )
193
194         self._test_bar.set_postfix(
195             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
196         )
197
198         self._train_bar.refresh()
199         self._test_bar.refresh()
200
201         self._train_bar.update(0)
202         self._test_bar.update(0)
203
204         self._train_bar.set_postfix(
205             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
206         )
207
208         self._test_bar.set_postfix(
209             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
210         )
211
212         self._train_bar.refresh()
213         self._test_bar.refresh()
214
215         self._train_bar.update(0)
216         self._test_bar.update(0)
217
218         self._train_bar.set_postfix(
219             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
220         )
221
222         self._test_bar.set_postfix(
223             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
224         )
225
226         self._train_bar.refresh()
227         self._test_bar.refresh()
228
229         self._train_bar.update(0)
230         self._test_bar.update(0)
231
232         self._train_bar.set_postfix(
233             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
234         )
235
236         self._test_bar.set_postfix(
237             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
238         )
239
240         self._train_bar.refresh()
241         self._test_bar.refresh()
242
243         self._train_bar.update(0)
244         self._test_bar.update(0)
245
246         self._train_bar.set_postfix(
247             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
248         )
249
250         self._test_bar.set_postfix(
251             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
252         )
253
254         self._train_bar.refresh()
255         self._test_bar.refresh()
256
257         self._train_bar.update(0)
258         self._test_bar.update(0)
259
260         self._train_bar.set_postfix(
261             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
262         )
263
264         self._test_bar.set_postfix(
265             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
266         )
267
268         self._train_bar.refresh()
269         self._test_bar.refresh()
270
271         self._train_bar.update(0)
272         self._test_bar.update(0)
273
274         self._train_bar.set_postfix(
275             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
276         )
277
278         self._test_bar.set_postfix(
279             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
280         )
281
282         self._train_bar.refresh()
283         self._test_bar.refresh()
284
285         self._train_bar.update(0)
286         self._test_bar.update(0)
287
288         self._train_bar.set_postfix(
289             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
290         )
291
292         self._test_bar.set_postfix(
293             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
294         )
295
296         self._train_bar.refresh()
297         self._test_bar.refresh()
298
299         self._train_bar.update(0)
300         self._test_bar.update(0)
301
302         self._train_bar.set_postfix(
303             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
304         )
305
306         self._test_bar.set_postfix(
307             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
308         )
309
310         self._train_bar.refresh()
311         self._test_bar.refresh()
312
313         self._train_bar.update(0)
314         self._test_bar.update(0)
315
316         self._train_bar.set_postfix(
317             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
318         )
319
320         self._test_bar.set_postfix(
321             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
322         )
323
324         self._train_bar.refresh()
325         self._test_bar.refresh()
326
327         self._train_bar.update(0)
328         self._test_bar.update(0)
329
330         self._train_bar.set_postfix(
331             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
332         )
333
334         self._test_bar.set_postfix(
335             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
336         )
337
338         self._train_bar.refresh()
339         self._test_bar.refresh()
340
341         self._train_bar.update(0)
342         self._test_bar.update(0)
343
344         self._train_bar.set_postfix(
345             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
346         )
347
348         self._test_bar.set_postfix(
349             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
350         )
351
352         self._train_bar.refresh()
353         self._test_bar.refresh()
354
355         self._train_bar.update(0)
356         self._test_bar.update(0)
357
358         self._train_bar.set_postfix(
359             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
360         )
361
362         self._test_bar.set_postfix(
363             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
364         )
365
366         self._train_bar.refresh()
367         self._test_bar.refresh()
368
369         self._train_bar.update(0)
370         self._test_bar.update(0)
371
372         self._train_bar.set_postfix(
373             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
374         )
375
376         self._test_bar.set_postfix(
377             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
378         )
379
380         self._train_bar.refresh()
381         self._test_bar.refresh()
382
383         self._train_bar.update(0)
384         self._test_bar.update(0)
385
386         self._train_bar.set_postfix(
387             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
388         )
389
390         self._test_bar.set_postfix(
391             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
392         )
393
394         self._train_bar.refresh()
395         self._test_bar.refresh()
396
397         self._train_bar.update(0)
398         self._test_bar.update(0)
399
400         self._train_bar.set_postfix(
401             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
402         )
403
404         self._test_bar.set_postfix(
405             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
406         )
407
408         self._train_bar.refresh()
409         self._test_bar.refresh()
410
411         self._train_bar.update(0)
412         self._test_bar.update(0)
413
414         self._train_bar.set_postfix(
415             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
416         )
417
418         self._test_bar.set_postfix(
419             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
420         )
421
422         self._train_bar.refresh()
423         self._test_bar.refresh()
424
425         self._train_bar.update(0)
426         self._test_bar.update(0)
427
428         self._train_bar.set_postfix(
429             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
430         )
431
432         self._test_bar.set_postfix(
433             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
434         )
435
436         self._train_bar.refresh()
437         self._test_bar.refresh()
438
439         self._train_bar.update(0)
440         self._test_bar.update(0)
441
442         self._train_bar.set_postfix(
443             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
444         )
445
446         self._test_bar.set_postfix(
447             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
448         )
449
450         self._train_bar.refresh()
451         self._test_bar.refresh()
452
453         self._train_bar.update(0)
454         self._test_bar.update(0)
455
456         self._train_bar.set_postfix(
457             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
458         )
459
460         self._test_bar.set_postfix(
461             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
462         )
463
464         self._train_bar.refresh()
465         self._test_bar.refresh()
466
467         self._train_bar.update(0)
468         self._test_bar.update(0)
469
470         self._train_bar.set_postfix(
471             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
472         )
473
474         self._test_bar.set_postfix(
475             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
476         )
477
478         self._train_bar.refresh()
479         self._test_bar.refresh()
480
481         self._train_bar.update(0)
482         self._test_bar.update(0)
483
484         self._train_bar.set_postfix(
485             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
486         )
487
488         self._test_bar.set_postfix(
489             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
490         )
491
492         self._train_bar.refresh()
493         self._test_bar.refresh()
494
495         self._train_bar.update(0)
496         self._test_bar.update(0)
497
498         self._train_bar.set_postfix(
499             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
500         )
501
502         self._test_bar.set_postfix(
503             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
504         )
505
506         self._train_bar.refresh()
507         self._test_bar.refresh()
508
509         self._train_bar.update(0)
510         self._test_bar.update(0)
511
512         self._train_bar.set_postfix(
513             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
514         )
515
516         self._test_bar.set_postfix(
517             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
518         )
519
520         self._train_bar.refresh()
521         self._test_bar.refresh()
522
523         self._train_bar.update(0)
524         self._test_bar.update(0)
525
526         self._train_bar.set_postfix(
527             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
528         )
529
530         self._test_bar.set_postfix(
531             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
532         )
533
534         self._train_bar.refresh()
535         self._test_bar.refresh()
536
537         self._train_bar.update(0)
538         self._test_bar.update(0)
539
540         self._train_bar.set_postfix(
541             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
542         )
543
544         self._test_bar.set_postfix(
545             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
546         )
547
548         self._train_bar.refresh()
549         self._test_bar.refresh()
550
551         self._train_bar.update(0)
552         self._test_bar.update(0)
553
554         self._train_bar.set_postfix(
555             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
556         )
557
558         self._test_bar.set_postfix(
559             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
560         )
561
562         self._train_bar.refresh()
563         self._test_bar.refresh()
564
565         self._train_bar.update(0)
566         self._test_bar.update(0)
567
568         self._train_bar.set_postfix(
569             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
570         )
571
572         self._test_bar.set_postfix(
573             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
574         )
575
576         self._train_bar.refresh()
577         self._test_bar.refresh()
578
579         self._train_bar.update(0)
580         self._test_bar.update(0)
581
582         self._train_bar.set_postfix(
583             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
584         )
585
586         self._test_bar.set_postfix(
587             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
588         )
589
590         self._train_bar.refresh()
591         self._test_bar.refresh()
592
593         self._train_bar.update(0)
594         self._test_bar.update(0)
595
596         self._train_bar.set_postfix(
597             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
598         )
599
600         self._test_bar.set_postfix(
601             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
602         )
603
604         self._train_bar.refresh()
605         self._test_bar.refresh()
606
607         self._train_bar.update(0)
608         self._test_bar.update(0)
609
610         self._train_bar.set_postfix(
611             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
612         )
613
614         self._test_bar.set_postfix(
615             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
616         )
617
618         self._train_bar.refresh()
619         self._test_bar.refresh()
620
621         self._train_bar.update(0)
622         self._test_bar.update(0)
623
624         self._train_bar.set_postfix(
625             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
626         )
627
628         self._test_bar.set_postfix(
629             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
630         )
631
632         self._train_bar.refresh()
633         self._test_bar.refresh()
634
635         self._train_bar.update(0)
636         self._test_bar.update(0)
637
638         self._train_bar.set_postfix(
639             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
640         )
641
642         self._test_bar.set_postfix(
643             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
644         )
645
646         self._train_bar.refresh()
647         self._test_bar.refresh()
648
649         self._train_bar.update(0)
650         self._test_bar.update(0)
651
652         self._train_bar.set_postfix(
653             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
654         )
655
656         self._test_bar.set_postfix(
657             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
658         )
659
660         self._train_bar.refresh()
661         self._test_bar.refresh()
662
663         self._train_bar.update(0)
664         self._test_bar.update(0)
665
666         self._train_bar.set_postfix(
667             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
668         )
669
670         self._test_bar.set_postfix(
671             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
672         )
673
674         self._train_bar.refresh()
675         self._test_bar.refresh()
676
677         self._train_bar.update(0)
678         self._test_bar.update(0)
679
680         self._train_bar.set_postfix(
681             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
682         )
683
684         self._test_bar.set_postfix(
685             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
686         )
687
688         self._train_bar.refresh()
689         self._test_bar.refresh()
690
691         self._train_bar.update(0)
692         self._test_bar.update(0)
693
694         self._train_bar.set_postfix(
695             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
696         )
697
698         self._test_bar.set_postfix(
699             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
700         )
701
702         self._train_bar.refresh()
703         self._test_bar.refresh()
704
705         self._train_bar.update(0)
706         self._test_bar.update(0)
707
708         self._train_bar.set_postfix(
709             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
710         )
711
712         self._test_bar.set_postfix(
713             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
714         )
715
716         self._train_bar.refresh()
717         self._test_bar.refresh()
718
719         self._train_bar.update(0)
720         self._test_bar.update(0)
721
722         self._train_bar.set_postfix(
723             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
724         )
725
726         self._test_bar.set_postfix(
727             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
728         )
729
730         self._train_bar.refresh()
731         self._test_bar.refresh()
732
733         self._train_bar.update(0)
734         self._test_bar.update(0)
735
736         self._train_bar.set_postfix(
737             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
738         )
739
740         self._test_bar.set_postfix(
741             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
742         )
743
744         self._train_bar.refresh()
745         self._test_bar.refresh()
746
747         self._train_bar.update(0)
748         self._test_bar.update(0)
749
750         self._train_bar.set_postfix(
751             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
752         )
753
754         self._test_bar.set_postfix(
755             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
756         )
757
758         self._train_bar.refresh()
759         self._test_bar.refresh()
760
761         self._train_bar.update(0)
762         self._test_bar.update(0)
763
764         self._train_bar.set_postfix(
765             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
766         )
767
768         self._test_bar.set_postfix(
769             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
770         )
771
772         self._train_bar.refresh()
773         self._test_bar.refresh()
774
775         self._train_bar.update(0)
776         self._test_bar.update(0)
777
778         self._train_bar.set_postfix(
779             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
780         )
781
782         self._test_bar.set_postfix(
783             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
784         )
785
786         self._train_bar.refresh()
787         self._test_bar.refresh()
788
789         self._train_bar.update(0)
790         self._test_bar.update(0)
791
792         self._train_bar.set_postfix(
793             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
794         )
795
796         self._test_bar.set_postfix(
797             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
798         )
799
800         self._train_bar.refresh()
801         self._test_bar.refresh()
802
803         self._train_bar.update(0)
804         self._test_bar.update(0)
805
806         self._train_bar.set_postfix(
807             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
808         )
809
810         self._test_bar.set_postfix(
811             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
812         )
813
814         self._train_bar.refresh()
815         self._test_bar.refresh()
816
817         self._train_bar.update(0)
818         self._test_bar.update(0)
819
820         self._train_bar.set_postfix(
821             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
822         )
823
824         self._test_bar.set_postfix(
825             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
826         )
827
828         self._train_bar.refresh()
829         self._test_bar.refresh()
830
831         self._train_bar.update(0)
832         self._test_bar.update(0)
833
834         self._train_bar.set_postfix(
835             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
836         )
837
838         self._test_bar.set_postfix(
839             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
840         )
841
842         self._train_bar.refresh()
843         self._test_bar.refresh()
844
845         self._train_bar.update(0)
846         self._test_bar.update(0)
847
848         self._train_bar.set_postfix(
849             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
850         )
851
852         self._test_bar.set_postfix(
853             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
854         )
855
856         self._train_bar.refresh()
857         self._test_bar.refresh()
858
859         self._train_bar.update(0)
860         self._test_bar.update(0)
861
862         self._train_bar.set_postfix(
863             train_accy="0.00", train_loss="0.00", test_accy="0.00", test_loss="0.00"
864         )
865
866         self._test_bar.set_postfix(
867             train_accy="
```

## Data [1]

As can be seen in the screen shot of the implementation the authors provide details and automatic downloads in the implementation, thus they are all public. Based on this snippet of the paper (page 5):

### Experiment Setups

**Datasets.** We validate our method on the widely used benchmark of class continual learning **CIFAR10** (Krizhevsky, Hinton et al. 2009), **CIFAR100** (Krizhevsky, Hinton et al. 2009) and **ImageNet-100** (Deng et al. 2009). CIFAR-10 is a dataset containing colored images classified into 10 classes, which consists of 50,000 training samples and 10,000 testing samples of size 32 \* 32. CIFAR-100 comprises 50,000 training images with 500 images per class and 10,000 testing images with 100 images per class. ImageNet-100 is composed of 100 classes with 1300 images per class for training and 500 images per class for validation. ImageNet-100 resembles real-world scenes with a higher resolution of 256\*256.

Short descriptions are given, statistics provided, citations presented. Although all brief, since the data is made highly accessible this brings the cost to 1.

### Configuration [3]

The authors present the values of the models hyperparameters hardcoded in the implementation, and conduct a grid search of some of the parameters in figure five.

Thus all used values can be determined easily, but as the search strategy/budget is only specified for a small subset of the hyperparameters / datasets the value is still 3.

**Parameter Analysis.** To verify the robustness of DSGD, we conduct experiments on CIFAR100-20 with different hyper-parameters  $\gamma \in \{0.9, 0.95, 1, 1.5, 2\}$  in dynamic topology graph construction. The results are presented in Figure 5(a). It is evident that the performance changes are minimal across different values of  $\gamma$ .

```
14
15  init_epoch = 200
16  init_lr = 0.1
17  init_milestones = [60, 120, 170]
18  init_lr_decay = 0.1
19  init_weight_decay = 0.0005
20
21
22  epochs = 80
23  lrate = 0.1
24  milestones = [40, 70]
25  lrate_decay = 0.1
26  batch_size = 128
27  weight_decay = 2e-4
28  num_workers = 8
29
```

## Experimental Procedure [2]

The authors state the metrics and data splits for the experiments on page 5. The also state that the metric is averaging by the specified formula 7. However it is not specified if the experiments are (not) repeated under a certain procedure, implying single run results thus a cost of 2 as this must be checked (against the implementation for example).

Implementation Details. For CIFAR10, CIFAR100, and ImageNet-100 datasets, we separately train all 10, 100, and 100 classes gradually with 2, 10 and 10 classes per stage. We use a fixed memory size of 2,000 exemplars, assigning 500 samples to labeled data and the remaining 1,500 samples to unlabeled data under sparse annotations. For the semi-supervised setting, we follow ORDisCo to allocate a small number of labels for each class and adhere to the standard experiment setup for selecting the labeled data (Oliver et al. 2018). To simplify the notation, we denote the benchmark as “dataset-(number of labels/class)”. For example, CIFAR10-30 indicates CIFAR10 with 30 labeled samples per class. Please see the Appendix for more details.

**Baseline and Metrics.** For CIFAR-10 and CIFAR-100, we employ a modified ResNet-32 (He et al. 2016) as our feature extractor, and adopt the standard ResNet-18 (He et al. 2016) as the feature extractor for ImageNet-100. We follow the Methods section and apply iCaRL&Fix and DER&Fix as the baselines and maintain the same architecture.

Following previous research on continual learning (Yan, Xie, and He 2021), we compare the top-1 average incremental accuracy:

$$A = \frac{1}{T} \sum_{t=1}^T A_t, \quad (7)$$

where  $A_t$  is the incremental accuracy on the task  $t$  and is defined by  $A_t = \frac{1}{t} \sum_{i=1}^t a_{t,i}$ , where  $a_{t,i}$  is the accuracy on the test set of the  $i^{th}$  task after learning the  $t^{th}$  task.

# pTSE: A Mult-model Ensemble Method for Probabilistic Time Series Forecasting

IJCAI 2023, [link here](#)

## Implementation [10]

There are no implementation details stated in the paper. No links can be found, no practical details whatsoever (Such as libraries/frameworks/programming languages/OS). We can't find any figure overviews nor pseudo code, thus each details while have to be extracted with a lot of effort from the paper for reproduction.

pTSE is publicly available datasets to test the performance.

## Data [5]

In 4.1 the authors state synthetic data is used and describe how this is generated. Although the code for it is missing, the generation is in general well documented, thus only increasing the cost with 1.

In 4.2 they discuss real world data analysis, and state they use three benchmark datasets (benchmark here indicates publicly available). For solar energy they provide a direct link in the footnote. For traffic they provide a citation. For electricity not link nor citation is provided, making it very difficult to find this dataset. A brief description on the data is given, but more information would be useful to understand the tasks in the data sets. Thus since one is easily accessible (direct link) one only semi (citations but very few details) and one not at all, but the results presented are evenly distributed across the datasets (table 1), we average the increase of 3. Thus  $1 + 1 + 3 = \underline{5}$ .

## Configuration [2]

The authors do not give an implementation nor pseudo code, so we will have to extract the parameter/hyperparameters with some effort. It is stated in 4.2 that the two ensemble methods are set to the recommended values of the original baselines, but we only care about the parameters of the presented method so this is irrelevant.

The authors discuss a bandwidth parameter sigma in 2.4, but this is 'optimally set' by bootstrapping, thus not a user input value. They state they incorporate

### 4.1 Synthetic Data Analysis

We simulated random sequences governed by HMM structures. We set  $K = 3, 5, 10$  and  $T = 1000$ . The transition matrix  $\mathbf{A}$  is chosen by first generating a matrix of uniformly distributed random numbers and then normalizing the matrix to ensure the sum of elements of each row equals 1. The emission function  $f_k(o)$  for each state  $k$  is simply set to a Gaussian distribution as  $\mathcal{N}(0.2k, \sqrt{k} + 1)$ ,  $(k = 1, \dots, K)$ . For each set of  $\{K, T, \mathbf{A}, \mathbf{F}(o)\}$ , we run the simulation procedure for 100 times, where during each time, an initial distribution  $\pi^0$  is randomly chosen. The results are presented in Figure 3. The empirical probability,  $\hat{F}(\tau)$ , shows fast convergence to  $\pi^* \int_{-\infty}^{\tau} \mathbf{F}(o) do$ , after  $T = 50$  for all simulated datasets, regardless of the state number  $K$  or the transition matrix  $\mathbf{A}$ .

### 4.2 Real World Data Analysis

a non-parametric method in 2.4, suggesting their method could also be parameter free. In section 2.1, they state what an HMM needs as input, but no configurable parameters seem to be introduced here either. However in the introduction (and the conclusions) they state their method is ‘a semi-parametric method’, thus leaving some ambiguity (Are these configurable parameters or learned parameters?). Thus as the implementation is missing, some effort will have to be made by the independent investigators to verify that indeed no parameters are needed for the method, increasing the cost by 1 to a total of **2**.

## Experimental Procedure [3]

The authors state the metrics used briefly in 4.2, and cite a previous work for it (Salinas et al.). In table 1 we see single values on each data set, but the last value is ‘Average risk’. It is a bit unclear what this average is calculated over, but perhaps with domain specific expertise this would be more clear. This would have to be looked up, increasing the cost by 1. At the end of the first paragraph of section 4.2 the authors state ‘The model performance would be evaluated on a 7-day-horizon test set’, but its unclear what this test actually is. It could be that these are provided statically by the benchmarks, but this is not stated and would have to be looked up increasing the cost by 1. Thus the total cost is **3**.

---

Four of the most popular probabilistic forecasting models are selected as the member models: SimpleFeedForwardEstimator (SFF), Transformer, DeepAR, and TemporalFusion-Transformer (TFT). As in [Salinas *et al.*, 2019b], we use  $q$ -risk metrics (quantile loss) to quantify the accuracy of a  $q$ -th quantile of the predictive distribution. Table 1 presents 0.5-risk, 0.9-risk, and the average risk of the output corresponding to each method. We also present a comparison of pTSE