Specification of Continual Few-Shot Learning Tasks

1. Task Definitions

In continual few-shot learning (CFSL), a task consists of a sequence of (training) support sets $\mathcal{G} = \{\mathcal{S}_j\}_{j=1}^{J}$, and a single (evaluation) target set $\mathcal{T}$. A support set is a set of input-output pairs $\mathcal{S}_j = \{(x_S^j, y_S^j)\}$ belonging to a number of different classes. A target set is a set of previously unseen input-output pairs $\mathcal{T} = \{(x_T, y_T)\}$ belonging to the same classes as in $\mathcal{G}$ such that $y_T = \bigcup_{j=0}^{J} y_S^j$.

A CFSL task is controlled by the following hyperparameters:

1. *Number of Support Sets Per Task* (NSS): the cardinality of $\mathcal{G}$, i.e. J
2. *Number of Classes Per Support Set* (N): the number of unique classes within each support set
3. *Number of Samples Per Support Set Class* (K): the number of data-points to be sampled for each class in the support set
4. *Number of Samples Per Target Set Class* (J): the number of data-points to be sampled for each class in the target set
5. *Class Change Interval* (CCI): the number of support sets sampled from the same class source, before that class source is resampled
6. *Overwrite* (O): a boolean variable that describes whether classes sampled at each support set will overwrite the class labels of the previously sampled support set (TRUE), or whether they will assigned new unique labels (FALSE)

The process of generating CFSL tasks is described in Algorithm 1. The exact hyperparameter combinations have been left out of this document as we allow freedom in setting arbitrary values to account for the wide research interest. However, we strongly encourage others to use the hyperparameter values that are inline with the existing literature.

Algorithm 1 Sampling a Continual Few-Shot Task

Data: Given labeled dataset $\mathcal{D}$, Number of support sets per task J, Number of classes per support set N, Number of samples per support set class K, Number of samples per class for target set M, Class change interval CCI, and class overwrite parameter O

$a = 0, b = 0$
$\mathcal{H} = \{\}$
for $a < (J/CCI)$ **do**
  Sample N classes from $\mathcal{D}$ excluding classes already in $\mathcal{H}$;
  Store classes sampled in task class cache $\mathcal{H}$
  if $O =$ *TRUE* **then**
    Assign labels 0 to N for the classes
  else
    Assign labels $a \times N$ to $(a+1) \times N$ for the classes
  end
  for $b < CCI$ **do**
    $j \leftarrow s \times CCI + b$
    Sample $K + M$ samples for each of the N classes
    Build support $\mathcal{S}_j$ with K samples per class
    Build target $\mathcal{T}_j$ with M samples per class
    Store sets $\mathcal{S}_j$ and $\mathcal{T}_j$
  end
end
Combine all target sets $\mathcal{T} = \bigcup_{j=0}^{J} \mathcal{T}_j$
Return $(\mathcal{S}_{0\ldots J}, \mathcal{T})$

2. Data Flow Dynamics

We restrict how a CFSL algorithm is allowed to process data in a given task. The model can only access one support set at a time for the purposes of knowledge extraction. Once this extraction has been completed, the current support set is deleted. Task knowledge can be stored within a parameter vector/matrix or an embedding vector/matrix. Once knowledge has been extracted by all the support sets, the model is tasked with predicting the classes of previously unseen samples in the target set. A generalization measure can be obtained by using the labels of the target set, once the model has produced its predictions to compute a task-level generalization measure.

3. Metrics

We define a set of useful metrics used to evaluate the methods.

1. Test Generalization Performance: a proposed model should be evaluated on at least the test sets of Omniglot and SlimageNet, on all of the task types of interest. This is done by presenting the model with a number of previously unseen continual tasks sampled from these

test sets, and then using the target set metrics as the task-level generalization metrics. To obtain a measure of generalization across the whole test set the model should be evaluated on *600* unique tasks and then take the mean and standard deviation of both the accuracy and cross-entropy performance of the model. These should be used as generalization measures to compare the model to other models.

2. Across-Task Memory (ATM): since the knowledge storage vectors that store support set knowledge have unrestricted memory, we also incorporate a metric that explicitly measures how memory efficient a certain model is, which can help differentiate between models of equal generalization performance. This measure can be computed by M/T where M is the total knowledge memory size across a whole task, and T is the total size of all samples in all the support sets within a task.

3. Computational Cost/Inference Memory: this metric measures the computational expense of the joint learner + model operations, as well as the memory footprint. This memory footprint is different than ATM, as ATM measures how much information is kept from each task when the next task is observed, whereas the inference memory footprint measures the memory footprint that the model itself needs to execute one cycle of inference + learning. Both of these costs can be approximated as the total amount of MAC (Multiply Accumulate) operations that a given model executes.

4. CSFL Rules

1. A task with $CCI = 1$, $NSS = 1$ will generate the same task type no matter what Overwrite is set to.

2. When classes are resampled for a new support set, assuming the CCI interval has been reached, the classes should be unique classes that have not appeared in any of the preceeding support sets within the current continual task.

3. When new samples are sampled, they should be unique samples, that have not been used in any other support set of the current continual task.

4. For SlimageNet the splits should be the exact splits that appear within train, val, and test in https://zenodo.org/record/3672132.

5. The smaller the ATM of a given model the more memory efficient it is. Maximal efficiency is achieved at 0, where no memory is used, and minimum efficiency is reached at infinity, where the model has infinite memory. A model that can store the whole observed dataset will have $ATM = 1$, whereas a model that stores 10 of the information in a data-point will have an ATM=0.1.