Wandering Within a World: Online Contextualized Few-Shot Learning

Mengye Ren\textsuperscript{1,3,5}  Michael L. Iuzzolino\textsuperscript{2}  Michael C. Mozer\textsuperscript{2,4}  Richard S. Zemel\textsuperscript{1,3,6}

\textsuperscript{1}University of Toronto  \textsuperscript{2}University of Colorado, Boulder  \textsuperscript{3}Vector Institute  \textsuperscript{4}Google Research  \textsuperscript{5}Uber ATG  \textsuperscript{6}CIFAR

\{mren,zemel\}@cs.toronto.edu, \{michael.iuzzolino,mozer\}@colorado.edu

Abstract

We aim to bridge the gap between typical human and machine-learning environments by extending the standard framework of few-shot learning to an online, continual setting. In this setting, episodes do not have separate training and testing phases, and instead models are evaluated online while learning novel classes. As in the real world, where the presence of spatiotemporal context helps us retrieve learned skills in the past, our online few-shot learning setting also features an underlying context that changes throughout time. Object classes are correlated within a context and inferring the correct context can lead to better performance. Building upon this setting, we propose a new few-shot learning dataset based on large scale indoor imagery that mimics the visual experience of an agent wandering within a world. Furthermore, we convert popular few-shot learning approaches into online versions and we also propose a new \textit{contextual prototypical memory} model that can make use of spatiotemporal contextual information from the recent past.\textsuperscript{1}

1 Introduction

In machine learning, many paradigms exist for training and evaluating models: standard train-then-evaluate, few-shot learning, incremental learning, continual learning, and so forth. None of these paradigms well approximates the naturalistic conditions that humans and artificial agents encounter as they wander within a physical environment. Consider, for example, learning and remembering peoples’ names in the course of daily life. We tend to see people in a given environment—work, home, gym, etc. We tend to repeatedly revisit those environments, with different environment base rates, nonuniform environment transition probabilities, and nonuniform base rates of encountering a given person in a given environment. We need to recognize when we do not know a person, and we need to learn to recognize them the next time we encounter them. We are not always provided with a name, but we can learn in a semi-supervised manner. And every training trial is itself an evaluation trial as we repeatedly use existing knowledge and acquire new knowledge. In this article, we propose a novel paradigm, \textit{online contextualized few-shot learning}, that approximates these naturalistic conditions, and we develop deep-learning architectures well suited for this paradigm.

In traditional few-shot learning (FSL)\textsuperscript{30,53}, training is episodic. Within an isolated episode, a set of new classes is introduced with a limited number of labeled examples per class—the \textit{support} set—followed by evaluation on an unlabeled \textit{query} set. While this setup has inspired the development of a multitude of meta-learning algorithms which can be trained to rapidly learn novel classes with a few labeled examples, the algorithms are focused solely on the few classes introduced in the current episode; the classes learned are not carried over to future episodes. Although incremental learning

\textsuperscript{1}Our code and dataset are released at: https://github.com/renmengye/oc-fewshot-public

Preprint. Under review.
Figure 1: **Online contextualized few-shot learning.** A) Our setup is similar to online learning, where there is no separate testing phase; model training and evaluation happen at the same time. The input at each time step is an (image, class-label) pair. The number of classes grows incrementally and the agent is expected to answer “new” for items that have not yet been assigned labels. Sequences can be semi-supervised; here the label is not revealed for every input item (labeled/unlabeled shown by red solid/grey dotted boxes). The agent is evaluated on the correctness of all answers. The model obtains learning signals only on labeled instances, and is correct if it predicts the label of previously-seen classes, or ‘new’ for new ones. B) The overall sequence switches between different learning environments. While the environment ID is hidden from the agent, inferring the current environment can help solve the task.

and continual learning methods [43][18] address the case where classes are carried over, the episodic construction of these frameworks seems artificial: in our daily lives, we do not learn new objects by grouping them with five other new objects, process them together, and then move on.

To break the rigid, artificial structure of continual and few-shot learning, we propose a new continual few-shot learning setting where environments are revisited and the total number of novel object classes increases over time. Crucially, model evaluation happens on each trial, very much like the setup in online learning. When encountering a new class, the learning algorithm is expected to indicate that the class is “new,” and it is then expected to recognize subsequent instances of the class once a label has been provided.

When learning continually in such a dynamic environment, contextual information can guide learning and remembering. Any structured sequence provides temporal context: the instances encountered recently are predictive of instances to be encountered next. In natural environments, spatial context—information in the current input weakly correlated with the occurrence of a particular class—can be beneficial for retrieval as well. For example, we tend to see our boss in an office setting, not in a bedroom setting. Human memory retrieval benefits from both spatial and temporal context [19][22].

In our online few-shot learning setting, we provide spatial context in the presentation of each instance and temporal structure to sequences, enabling an agent to learn from both spatial and temporal context. Besides developing and experimenting on a toy benchmark using handwritten characters [30], we also propose a new large-scale benchmark for online contextualized few-shot learning derived from indoor panoramic imagery [7]. In the toy benchmark, temporal context can be defined by the co-occurrence of character classes. In the indoor environment, the context—temporal and spatial—is a natural by-product as the agent wandering in between different rooms.

We propose a model that can exploit contextual information, called contextual prototypical memory (CPM), which incorporates an RNN to encode contextual information and a separate prototype memory to remember previously learned classes (see Figure 4). This model obtains significant gains on few-shot classification performance compared to models that do not retain a memory of the recent past. We compare to classic few-shot algorithms [53][2][50][20][46] extended to an online setting, and CPM consistently achieves the best performance.

The main contributions of this paper are as follows. First, we define an online contextualized few-shot learning (OC-FSL) setting to mimic naturalistic human learning. Second, we build two datasets. The RoamingOmniglot dataset is based on handwritten characters from Omniglot [30] and the RoamingRooms dataset is our new few-shot learning dataset based on indoor imagery [7], which resembles the visual experience of a wandering agent. Third, we benchmark classic FSL methods and also explore our CPM model, which combines the strengths of RNNs for modeling temporal context and Prototypical Networks [50] for memory consolidation and rapid learning.
Table 1: Comparison of past FSL and CL paradigms vs. our online contextualized FSL (OC-FSL).

| Tasks                  | Few Shot | Semi-sup. | Continual | Online | Predict | New | Soft Context | Switch |
|------------------------|----------|-----------|------------|--------|---------|-----|---------------|--------|
| Incremental Learning (IL) | [43]    |     ●     |   ●        | ○      | ○       | ○   | ○             | ○      |
| Few-shot (FSL)         | [43]    |     ●     |   ●        | ○      | ○       | ○   | ○             | ○      |
| Incremental FSL        | [44]    |     ●     |   ●        | ○      | ○       | ○   | ○             | ○      |
| Cls. Incremental FSL   | [43]    |     ●     |   ●        | ○      | ○       | ○   | ○             | ○      |
| Semi-sup. FSL          | [45]    |     ●     |   ●        | ○      | ○       | ○   | ○             | ○      |
| MOCA                   | [15]    |     ●     |   ●        | ○      | ○       | ○   | ○             | ○      |
| Online Mixture         | [21]    |     ●     |   ●        | ○      | ○       | ○   | ○             | ○      |
| Online Meta            | [20]    |     ●     |   ●        | ○      | ○       | ○   | ○             | ○      |
| Continual FSL          | [4]     |     ●     |   ●        | ○      | ○       | ○   | ○             | ○      |
| OSAKA*                 | [5]     |     ●     |   ●        | ○      | ○       | ○   | ○             | ○      |
| OC-FSL (Ours)          |         |           |            | ○      | ○       | ○   | ○             | ○      |

* denotes concurrent work.

2 Related Work

In this section, we briefly review paradigms that have been used for few-shot learning (FSL) and continual learning (CL). Table 1 compares these paradigms based on various properties of the task; to denote properties that are incorporated in some preliminary form, we use ● to denote properties that are not fully implemented but have some preliminary form. For example, the class-incremental learning paradigm evaluates models after each task is completed, which is similar to our online evaluation in spirit but still not the same as the evaluation does not take place after each example. Our proposed online contextual few-shot learning (OC-FSL) spans the complete set of features of the other paradigms. We also review relevant models and their relationship to our CPM.

**Few-shot Learning:** FSL [30, 32, 27, 53] considers learning new tasks with few labeled examples. FSL models can be categorized as based on: metric learning [53, 50], memory [46], and gradient adaptation [10, 34]. The model we propose, CPM, lies on the boundary between these approaches, as we use an RNN to model the temporal context but we also use metric-learning mechanisms and objectives to train.

Several previous efforts have aimed to extend few-shot learning to incorporate more natural constraints. One such example is semi-supervised FSL [45], where models learn not only from a few labeled examples but also from a pool of unlabeled examples. While traditional FSL only tests the learner on novel classes, *incremental FSL* [13, 44] tests on novel classes together with a set of base classes. These approaches, however, have not explored how to iteratively add new classes.

In concurrent work, Antoniou et al. [4] extend FSL to a continual setting based on image sequences, each of which is divided into stages with a fixed number of examples per class followed by a query set. Focuses on more flexible and faster adaptation since the models are evaluated online, and context is a soft constraint instead of a hard separation of tasks. Moreover, new classes need to be identified as part of the sequence, crucial to any learner’s incremental acquisition of knowledge.

**Continual Learning:** Continual (or lifelong) learning is a parallel line of research that aims to handle a sequence of dynamic tasks [26, 33, 36, 55]. A key challenge here is catastrophic forgetting [39, 12], where the model “forgets” a task that has been learned in the past. Incremental learning [43, 44] is a form of continual learning, where each task is an incremental batch of several new classes. This assumption that novel classes always come in batches seems unnatural.

Traditionally, continual learning is studied with tasks such as permuted MNIST [31] or split-CIFAR [28]. Recent datasets aim to consider more realistic continual learning, such as CORRe50 [35] and OpenLORIS [49]. We summarize core features of these continual learning datasets in Appendix A. First, both CORRe50 and OpenLORIS have relatively few object classes, which makes meta-learning approaches inapplicable; second, both contain images of small objects with minimal occlusion and viewpoint changes; and third, OpenLORIS does not have the desired incremental class learning.

In concurrent work, [5] proposes a setup to unify continual learning and meta-learning with a similar online evaluation procedure. However, there are several notable differences. First, their models focus on a general loss function without a specific design for predicting new classes; they predict new tasks by examining if the loss of the current prediction exceeds some threshold. Second, the sequences of inputs are fully supervised, which allows their models to evaluate the loss function...
when it indicates that the item has yet to be assigned a label. Working memory that can retain novel objects and spatiotemporal contexts. Lastly, the prototype weights slowly to learn a more robust representation. The hippocampal system can be further divided into an episodic memory for individual spatiotemporal events and a semantic memory for a concrete piece of knowledge. Our proposed CPM contains parallels to these components. Long term statistical learning is captured in a CNN that produces a deep embedding. An RNN holds a type of working memory that can retain novel objects and spatiotemporal contexts. Lastly, the prototype memory represents the semantic memory, which consolidates multiple events into a single knowledge vector. Other deep learning researchers have proposed multiple memory systems for continual learning. In [42], the learning algorithm is heuristic and representations come from pretrained networks. In [24], a prototype memory is used for recalling recent examples and rehearsal from a generative model allows this knowledge to be integrated and distilled into a long-term memory.

### 3 Online Contextualized Few-Shot Learning

In this section, we introduce our new online contextualized few-shot learning (OC-FSL) setup in the form of a sequential decision problem, and then introduce our new benchmark datasets.

**Continual few-shot classification as a sequential decision problem:** In traditional few-shot learning, an episode is constructed by a support set \( S \) and a query set \( Q \). A few-shot learner \( f \) is expected to predict the class of each example in the query set \( x^Q \) based on the support set information: \( \hat{y}^Q = f(x^Q; (x_1^S, y_1^S), \ldots, (x_N^S, y_N^S)) \). This setup is not a natural fit for continual learning, since it is unclear when to insert a query set into the sequence.

Inspired by the online learning literature, we can convert continual few-shot learning into a sequential decision problem, where every input example is also part of the evaluation: \( \hat{y}_t = f(x_t; (x_1, y_1), \ldots, (x_{t-1}, \tilde{y}_{t-1})) \), for \( t = 1 \ldots T \), where \( \tilde{y} \) here further allows that the sequence of inputs to be semi-supervised: \( \tilde{y} \) equals \( y_t \) if labeled, or otherwise \(-1\). The setup in [46] and [23] is similar; they train RNNs using such a temporal sequence as input. However, their evaluation relies on another "query set" at the end of the sequence. We instead evaluate online while learning.

**Contextualized environments:** Typical continual learning consists of a sequence of tasks, and models are trained sequentially for each task. This feature is also preserved in many incremental learning settings [43]. For instance, the split-CIFAR task divides CIFAR-100 into 10 learning environments, each with 10 classes, presented sequentially. In our formulation, the sequence returns to earlier environments (see Figure 1-B), which enables assessment of long-term durability of knowledge. Although the ground-truth environment identity is not provided, we structure the task such that the environment itself provides contextual cues which can constrain the correct class label. Spatial cues in the input help distinguish one environment from another. Temporal cues are implicit because the sequence tends to switch environments infrequently, allowing recent inputs to be beneficial in guiding the interpretation of the current input.
RoamingOmniglot: The Omniglot dataset \cite{lake2017building} contains 1623 handwritten characters from 50 different alphabets. We split the alphabets into 31 for training, 5 for validation, and 13 for testing. We augment classes by 90 degree rotations to create 6492 classes in total. Each contextualized few-shot learning image sequence contains 150 images, drawn from a random sample of 5-10 alphabets, for a total of 50 classes per sequence. These classes are randomly assigned to 5 different environments; within an environment, the characters are distributed according to a Chinese restaurant process \cite{levine2011bayesian} to mimic the imbalanced long-tail distribution of naturally occurring objects. We switch between environments using a Markov switching process; i.e., at each step there is a constant probability of switching to another environment. An example sequence is shown in Figure 2-A.

RoamingRooms: As none of the current few-shot learning datasets provides the natural online learning experience that we would like to study, we created our own dataset using simulated indoor environments. We formulate this as a few-shot instance learning problem, which could be a use case for a home robot: it needs to quickly recognize and differentiate novel object instances, and large viewpoint variations can make this task challenging (see examples in Figure 2-B). There are over 7,000 unique instance classes in the dataset, making it suitable to meta-learning approaches.

Our dataset is derived from the Matterport3D dataset \cite{chang2017matterport3d}, which has 90 indoor worlds captured using panoramic depth cameras. We split these into 60 worlds for training, 10 for validation and 20 for testing. We use MatterSim \cite{zeng2018matter} to load the simulated world and collect camera images and use HabitatSim \cite{chen2019habitat} to simulate 3D mesh and align instance segmentation labels onto 2D image space. We created a random walking agent to collect the virtual visual experience. For each viewpoint in the random walk, we randomly sample one object from the image sensor and highlight it with the available instance segmentation, forming an input frame. Each viewpoint provides environmental context—the other objects present in the room with the highlighted object.

Figure 3-A shows the object instance distribution. We see strong temporal correlation, as 30% of the time the same instance appears in the next frame (Figure 3-B), but there is also a significant proportion of revisits. On average, there are three different viewpoints per 100-image sequence (Figure 3-C). Details and other statistics of our proposed datasets are included in the Appendix.

Figure 2: Sample online contextualized few-shot learning sequences. A) RoamingOmniglot. Red solid boxes denote labeled examples of Omniglot handwritten characters, and dotted boxes denote unlabeled ones. Environments are shown in colored labels in the top left corner. B) Image frame samples of a few-shot learning sequence in our RoamingRooms dataset collected from a random walking agent. The task here is to recognize novel instances in the home environment. C) The growth of total number of labeled classes in a sequence for RoamingOmniglot (top) and RoamingRooms (bottom).

Figure 3: Statistics for our RoamingRooms dataset. Plots show a natural long tail distribution of instances grouped into categories. An average sequence has 3 different view points. Sequences are highly correlated in time but revisits are not uncommon.
4 Contextual Prototypical Memory Networks

In the online contextualized few-shot learning setup, the few-shot learner can potentially improve by modeling the temporal context. Metric learning approaches [53,20] typically ignore temporal relations and directly compare the similarity between training and test samples. Gradient-based approaches [10,20], on the other hand, have the ability to adapt to new contexts, but they do not naturally handle new and unlabeled examples. We instead propose a simple yet effective approach that augments the classic Prototypical Network with a temporal contextual encoder using an RNN, shown in Figure 4. Next, we describe our approach in detail.

Prototype memory: We start describing our model with the prototype memory, which is an online version of the Prototypical Network (or ProtoNet) [50]. ProtoNet can be viewed as a knowledge base memory, where each object class $k$ is represented by a prototype vector $p_k$, computed as the mean vector of all the support instances of the class in a sequence. It can also be applied to our task of online few-shot learning naturally, with some modifications. Suppose that at time-step $t$ we have already stored a few classes in the memory, each represented by their current prototype $p_t[k]$, and we would like to query the memory using the input feature $h_t$. We model our prototype memory as

$$\hat{y}_{t,k} = \text{softmax}\left(-\frac{||h_t - p_t[k]||_2^2}{M_t}\right),$$

where $||\cdot||_{M_t}$ is the squared Mahalanobis distance for some to-be-specified, time-varying hyperparameter matrix $M_t$. To predict whether an example is of a new class, we can use a separate novelty output $\hat{u}_t^w$ with sigmoid activation, similar to the approach introduced in [45], where $\beta_t^w$ and $\gamma_t^w$ are yet-to-be-specified thresholding hyperparameters (the superscript $r$ stands for read):

$$\hat{u}_t^w = \text{sigmoid}\left(\frac{\min_k ||h_t - p_t[k]||_2^2 - \beta_t^w}{\gamma_t^w}\right).$$

Memory consolidation with online prototype averaging: Traditionally, ProtoNet uses the average representation of a class across all support examples. Here, we must be able to adapt the prototype memory incrementally at each step. Fortunately, we are able to recover the same prototypes as a regular ProtoNet computes offline by using online averaging. For each prototype $k$, we store a count scalar $c[k]_t$ to indicate the number of examples that have been added to this prototype up to time $t$. When the current example is unlabeled, $y_t$ is encoded as $-1$, and the model’s own prediction $\hat{y}_t^w$ will determine which prototype to update; in this case, the model must also determine a strength of belief, $\hat{u}_t^w$, that the current unlabeled example should be treated as a new class. Given $\hat{u}_t^w$ and $y_t$, the model can then update a prototype:

$$\hat{u}_t^w = \text{sigmoid}\left(\min_k ||h_t - p_t[k]||_2^2 - \beta_t^w / \gamma_t^w\right),$$

$$\Delta [k]_t = \begin{cases} 1[y_t = k] + \hat{y}_t(1 - \hat{u}_t^w)/\gamma_t^w & \text{if } \Delta [k]_t > 0, \\ 1[y_t = -1] & \text{otherwise}. \end{cases}$$

$$c[k]_t = c[k]_{t-1} + \Delta [k]_t,$$

$$p[k]_t = \frac{1}{c[k]_t} \left( p[k]_{t-1} c[k]_{t-1} + h_t \Delta [k]_t \right)$$

As-yet-unspecified hyperparameters $\beta_t^w$ and $\gamma_t^w$ are required (the superscript $w$ is for write). These parameters for the online-updating novelty output $\hat{u}_t^w$ are distinct from $\beta_r^w$ and $\gamma_r^w$ in Equation (3). The intuition is that for “self-teaching” to work, the model potentially needs to be more conservative in creating new classes (avoiding corruption of prototypes) than in predicting an input as being a new class.
Contextual RNN: Instead of directly using the features from the CNN $h_t^{\text{CNN}}$ as input features to the prototype memory, we would also like to use contextual information from the recent past. Above we introduced threshold hyperparameters $\beta^r_t$, $\gamma^r_t$, $\beta^w_t$, $\gamma^w_t$ as well as the metric parameter $M_t$. We let the contextual RNN output these additional control parameters, so that the unknown thresholds and metric function can adapt based on the information in the context. The RNN produces the context vector $h_t^{\text{RNN}}$ and other control parameters conditioned on $h_t^{\text{CNN}}$:
\[
[z_t, h_t^{\text{RNN}}, m_t, \beta^r_t, \gamma^r_t, \beta^w_t, \gamma^w_t] = \text{RNN}(h_t^{\text{CNN}}; z_{t-1}),
\] (7)

where $z_t$ is the recurrent state of the RNN, and $m_t$ is the diagonal vector of $M_t$. The context, $h_t^{\text{RNN}}$, serves as an additive bias on the state vector used for FSL: $h_t = h_t^{\text{CNN}} + h_t^{\text{RNN}}$.

Loss function: The loss function is computed after an entire sequence ends and all network parameters are learned end-to-end. The loss is composed of two parts. The first is binary cross-entropy (BCE), for telling whether each example has been assigned a label or not, i.e., prediction of new classes. Second we use a multi-class cross-entropy for classifying among the known ones. We can write down the overall loss function as follows:
\[
\mathcal{L} = \frac{1}{T} \sum_{t=1}^{T} \lambda \left[ -\mathbb{1}[y_t < 0] \log(\hat{u}_t^-) - \mathbb{1}[y_t \geq 0] \log(1 - \hat{u}_t^+) \right] + \sum_{k=1}^{K} \mathbb{1}[y_t = k] \log(y_{t,k}).
\] (8)

5 Experiments

In this section, we show experimental results for our online contextualized few-shot learning paradigm, using RoamingOmniglot and RoamingRooms (see Sec. 3) to evaluate our model, CPM, and other state-of-the-art methods. For Omniglot, we apply an 8 × 8 CutOut [9] to each image to make the task more challenging. Details about the split information can be found in Appendix A.

Implementation details: For the RoamingOmniglot experiment we used the common 4-layer CNN for few-shot learning with 64 channels in each layer. For the RoamingRooms experiment we resize the input to 120 × 160 and we use the ResNet-12 architecture [41]. To represent the feature of the input image with an attention mask, we concatenate the global average pooled feature with the attention ROI feature, resulting in a 512d feature vector. For the contextual RNN, in both experiments we used an LSTM [17] with a 256d hidden state. We include additional training details in Appendix B.

Evaluation metrics: In order to compute a single number that characterizes the learning ability over sequences, we propose to use average precision (AP) to combine the prediction on old and...
Figure 5: **Few-shot classification accuracy over time.** **Left:** RoamingOmniglot. **Right:** RoamingRooms. **Top:** Supervised. **Bottom:** Semi-supervised. An offline logistic regression (Offline LR) baseline is also included, using pretrained ProtoNet features. It is trained on all labeled examples except for the one at the current time step.

Figure 6: **Effect of spatiotemporal context.** Spatiotemporal context are added separately and together in RoamingOmniglot, by introducing texture background and temporal correlation. **Left:** Stimuli used for spatial cue of the background environment. **Right:** Our CPM model significantly benefits from the presence of a temporal context (“+Temporal” and “+Both”), while Spatial context helps both CPM and online ProtoNet.

new classes. Concretely, all predictions are sorted by their old vs. new scores, and we compute AP using the area under the precision-recall curve. True positive is defined as the correct prediction of a multi-class classification among known classes. We also compute the “N-shot” accuracy; i.e., the average accuracy after seeing the label N times in the sequence. Note that these accuracy scores only reflect the performance on known class predictions. All numbers are reported with an average over 2,000 sequences and for N-shot accuracy standard error is also included.

Comparisons: To evaluate the merits of our proposed model, we implement classic few-shot learning and online meta-learning methods. More implementation and training details of these baseline methods can be found in Appendix B. 

- **OML [20]:** This is an online version of MAML [10]. It performs one gradient descent step for each labeled input image, and slow weights are learned via backpropagation through time. 
- **LSTM [17] & DNC [14]:** We include RNN methods for comparison as well. Differentiable neural computer (DNC) is an improved version of memory augmented neural network (MANN) [46]. 
- **Online MatchingNet [53], IMP [2] & ProtoNet [50]:** We used the same negative Euclidean distance as the similarity function for these three metric learning based approaches. In particular,
MatchingNet stores all examples and performs nearest neighbor matching, which can be memory inefficient. Note that Online ProtoNet is a variant of our method without the contextual RNN.

**Main results:** Our main results are shown in Table 2 and 3, including both supervised and semi-supervised settings. Online ProtoNet is a direct comparison without our contextual RNN and it is clear that CPM is significantly better. Our method is slightly worse than Online MatchingNet in terms of 3-shot accuracy on the RoamingRooms semisupervised benchmark. This can be explained by the fact that MatchingNet stores all past seen examples, whereas CPM only stores one prototype per class. Per timestep accuracy is plotted in Figure 5, and the decaying accuracy is due to the increasing number of classes over time. In RoamingOmniglot, CPM is able to closely match or even sometimes surpass the offline classifier, which re-trains at each step and uses all images in a sequence except the current one. This is reasonable as our model is able to leverage information from the current context.

**Effect of spatiotemporal context:** To answer the question whether the gain in performance is due to spatiotemporal reasoning, we conduct the following experiment comparing CPM with online ProtoNet. We allow the CNN to have the ability to recognize the context in RoamingOmniglot by adding a texture background image using the Kylberg texture dataset [29] (see Figure 6 left). As a control, we can also destroy the temporal context by shuffling all the images in a sequence. We train four different models on dataset controls with or without the presence of spatial or temporal context, and results are shown in Figure 6. First, both online ProtoNet and CPM benefit from the inclusion of a spatial context. This is understandable as the CNN has the ability to learn spatial cues, which re-confirms our main hypothesis that successful inference of the current context is beneficial to novel object recognition. Second, only our CPM model benefits from the presence of temporal context, and it receives distinct gains from spatial and temporal contexts.

**Ablation studies:** We ablate each individual module we introduce. Results are shown in Tables 4 and 5. Table 4 studies different ways we use the RNN, including the context vector $h^{\text{RNN}}$, the predicted threshold parameters $\beta^t_*, \gamma^t_*$, and the predicted metric scaling vector $m_t$. Table 5 studies various ways to learn from unlabeled examples, where we separately disable the RNN update, prototype update, and distinct write-threshold parameters $\beta^w_*, \gamma^w_*$ (vs. using read-threshold parameters). We verify that each component has a positive impact on the performance.

**6 Conclusion**

We proposed online contextualized few-shot learning, OC-FSL, a paradigm for machine learning that emulates a human or artificial agent interacting with a physical world. It combines multiple properties to create a challenging learning task: every input must be classified or flagged as novel, every input is also used for training, semi-supervised learning can potentially improve performance, and the temporal distribution of inputs is non-IID and comes from a generative model in which input and class distributions are conditional on a latent environment with Markovian transition probabilities. We proposed the RoamingRooms dataset to simulate an agent wandering within a physical world. We also proposed a new model, CPM, which uses an RNN to extract spatiotemporal context from the input stream and to provide control settings to a prototype-based FSL model. In the context of naturalistic domains like RoamingRooms, CPM is able to leverage contextual information to attain performance unmatched by other state-of-the-art FSL methods.
Acknowledgments

We thank Fei Sha, James Lucas and Eleni Triantafillou for helpful feedback on an earlier draft of the manuscript. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute (www.vectorinstitute.ai/#partners). This project is supported by NSERC and the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DoI/IBC) contract number D16PC00003. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/IBC, or the U.S. Government.

References

[1] D. J. Aldous. Exchangeability and related topics. In École d’Été de Probabilités de Saint-Flour XIII—1983, pages 1–198. Springer, 1985.

[2] K. R. Allen, E. Shelhamer, H. Shin, and J. B. Tenenbaum. Infinite mixture prototypes for few-shot learning. In Proceedings of the 36th International Conference on Machine Learning, ICML, 2019.

[3] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sündenhauf, I. Reid, S. Gould, and A. van den Hengel. Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2018.

[4] A. Antoniou, M. Patacchiola, M. Ochal, and A. J. Storkey. Defining benchmarks for continual few-shot learning. CoRR, abs/2004.11967, 2020.

[5] M. Caccia, P. Rodríguez, O. Ostapenko, F. Normandin, M. Lin, L. Caccia, I. H. Laradji, I. Rish, A. Lacoste, D. Vázquez, and L. Charlin. Online fast adaptation and knowledge accumulation: a new approach to continual learning. CoRR, abs/2003.05856, 2020.

[6] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari. End-to-end incremental learning. In 15th European Conference on Computer Vision, ECCV, 2018.

[7] A. X. Chang, A. Dai, T. A. Funkhouser, M. Halber, M. Nießner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from RGB-D data in indoor environments. In International Conference on 3D Vision, 3DV, 2017.

[8] N. Cohen and L. Squire. Preserved learning and retention of pattern-analyzing skill in amnesia: dissociation of knowing how and knowing that. Science, 210(4466):207–210, 1980.

[9] T. Devries and G. W. Taylor. Improved regularization of convolutional neural networks with cutout. CoRR, abs/1708.04552, 2017.

[10] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the 34th International Conference on Machine Learning, ICML, 2017.

[11] C. Finn, A. Rajeswaran, S. M. Kakade, and S. Levine. Online meta-learning. In Proceedings of the 36th International Conference on Machine Learning, ICML, 2019.

[12] R. M. French. Catastrophic forgetting in connectionist networks. Trends in cognitive sciences, 3(4):128–135, 1999.

[13] S. Gidaris and N. Komodakis. Dynamic few-shot visual learning without forgetting. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2018.

[14] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, A. P. Badia, K. M. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis. Hybrid computing using a neural network with dynamic external memory. Nature, 538(7626):471–476, Oct 2016.

[15] J. Harrison, A. Sharma, C. Finn, and M. Pavone. Continuous meta-learning without tasks. CoRR, abs/1912.08866, 2019.
[16] J. He, R. Mao, Z. Shao, and F. Zhu. Incremental learning in online scenario. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, 2020.

[17] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.

[18] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin. Learning a unified classifier incrementally via rebalancing. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2019.

[19] M. W. Howard. Temporal and spatial context in the mind and brain. *Current opinion in behavioral sciences*, 17:14–19, 2017.

[20] K. Javed and M. White. Meta-learning representations for continual learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2019.

[21] G. Jerfel, E. Grant, T. Griffiths, and K. A. Heller. Reconciling meta-learning and continual learning with online mixtures of tasks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2019.

[22] M. J. Kahana. *Foundations of human memory*. Oxford University Press, New York, 2012. OCLC: ocn744297060.

[23] L. Kaiser, O. Nachum, A. Roy, and S. Bengio. Learning to remember rare events. In *5th International Conference on Learning Representations, ICLR*, 2017.

[24] R. Kemker and C. Kanan. Fearnet: Brain-inspired model for incremental learning. In *Proceedings of 6th International Conference on Learning Representations (ICLR)*, 2018.

[25] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*, 2015.

[26] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[27] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, 2015.

[28] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[29] G. Kylberg. *Kylberg Texture Dataset v. 1.0*. Centre for Image Analysis, Swedish University of Agricultural Sciences and . . . , 2011.

[30] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[31] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[32] F. Li, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Comput. Vis. Image Underst.*, 106(1):59–70, 2007.

[33] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(12):2935–2947, 2018.

[34] Z. Li, F. Zhou, F. Chen, and H. Li. Meta-sgd: Learning to learn quickly for few shot learning. *CoRR*, abs/1707.09835, 2017.

[35] V. Lomonaco and D. Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *1st Annual Conference on Robot Learning, CoRL*, 2017.

[36] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2017.

[37] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.
[38] J. McClelland, B. McNaughton, and R. O’Reilly. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. Psychological review, 102:419–57, 08 1995.

[39] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In Psychology of learning and motivation, volume 24, pages 109–165. Elsevier, 1989.

[40] M. Muja and D. Lowe. Flann-fast library for approximate nearest neighbors user manual. Computer Science Department, University of British Columbia, Vancouver, BC, Canada, 2009.

[41] B. N. Oreshkin, P. R. López, and A. Lacoste. TADAM: task dependent adaptive metric for improved few-shot learning. In Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS, 2018.

[42] G. I. Parisi, J. Tani, C. Weber, and S. Wermter. Lifelong learning of spatiotemporal representations with dual-memory recurrent self-organization. Frontiers in neurorobotics, 12:78–78, Nov 2018.

[43] S. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2017.

[44] M. Ren, R. Liao, E. Fetaya, and R. S. Zemel. Incremental few-shot learning with attention attractor networks. In Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS, 2019.

[45] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel. Meta-learning for semi-supervised few-shot classification. In 6th International Conference on Learning Representations, ICLR, 2018.

[46] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. P. Lillicrap. Meta-learning with memory-augmented neural networks. In Proceedings of the 33nd International Conference on Machine Learning, ICML, 2016.

[47] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A Platform for Embodied AI Research. In Proceedings of the IEEE/CVF International Conference on Computer Vision, ICCV, 2019.

[48] J. Schmidhuber. Evolutionary principles in self-referential learning, or on learning how to learn: The meta-meta-... hook. Diplomarbeit, Technische Universität München, München, 1987.

[49] Q. She, F. Feng, X. Hao, Q. Yang, C. Lan, V. Lomonaco, X. Shi, Z. Wang, Y. Guo, Y. Zhang, F. Qiao, and R. H. M. Chan. Openloris-object: A dataset and benchmark towards lifelong object recognition. CoRR, abs/1911.06487, 2019.

[50] J. Snell, K. Swersky, and R. S. Zemel. Prototypical networks for few-shot learning. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, NeurIPS, 2017.

[51] X. Tao, X. Hong, X. Chang, S. Dong, X. Wei, and Y. Gong. Few-shot class-incremental learning. CoRR, abs/2004.10956, 2020.

[52] S. Thrun. Lifelong learning algorithms. In Learning to learn, pages 181–209. Springer, 1998.

[53] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems, NeurIPS, 2016.

[54] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu. Large scale incremental learning. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2019.

[55] J. Yoon, E. Yang, J. Lee, and S. J. Hwang. Lifelong learning with dynamically expandable networks. In 6th International Conference on Learning Representations, ICLR, 2018.
A Dataset Details

A.1 RoamingOmniglot Details

For the RoamingOmniglot experiments, we use sequences with maximum 150 images, from 5 environments. For individual environment, we use a Chinese restaurant process to sample the class distribution. In particular, the probability of sampling a new class is:

\[ p_{\text{new}} = \frac{k\alpha + \theta}{m + \theta} \quad (9) \]

where \( k \) is the number of classes that we have already sampled in the environment, and \( m \) is the total number of instances we have in the environment. \( \alpha \) is set to 0.2 and \( \theta \) is set to 1.0 in all experiments.

The environment switching is implemented by a Markov switching process. At each step in the sequence there is a constant probability \( p_{\text{switch}} \) that switches to another environment. For all experiments, we set \( p_{\text{switch}} \) to 0.2. We truncate the maximum number of appearances per class to 6. If the maximum appearance is reached, we will sample another class.

A.2 Additional RoamingRooms Statistics

Statistics of the RoamingRooms are included in Table 6, in comparison to other few-shot and continual learning datasets. Note that since RoamingRooms is collected from a simulated environment, with 90 indoor worlds consisting of 1.2K panorama images and 1.22M video frames. The dataset contains about 6.9K random walk sequences with a maximum of 200 frames per sequence. For training we randomly crop 100 frames to form a training sequence. There are 7.0K unique instance classes.

Plots of additional statistics of RoamingRooms are shown in Figure 7. In addition to the ones shown in the main paper, instances and viewpoints also follow long tail distributions. The number of objects in each frame follows an exponential distribution.

A.3 RoamingRooms Simulator Details

We generate our episodes with a two-stage process using two simulators – HabitatSim [47] and MatterSim [3] – because HabitatSim is based on 3D meshes and using HabitatSim alone will result in poor image quality due to incorrect mesh reconstruction. Therefore we sacrificed the continuous movement of agents within HabitatSim and base our environment navigation on the discrete viewpoints in MatterSim, which is based on real panoramic images. The horizontal field of view is set to 90 degrees for HabitatSim and 100 degrees for MatterSim, and we simulate with 800×600 resolution.

The first stage of generation involves randomly picking a sequence of viewpoints on the connectivity graph within MatterSim. For each viewpoint, the agent scans the environment along the yaw and pitch axes for a random period of time until a navigable viewpoint is within view. The time spent in a single viewpoint follows a Gaussian distribution with mean 5.0 and standard deviation 1.0. At the start of each new viewpoint, the agent randomly picks a direction to rotate and takes 12.5 degree steps along the yaw axis, and with 95\% probability, a 5 degree rotation along the pitch axis is applied in a randomly chosen direction. When a navigable viewpoint is detected, the agent will navigate to the new viewpoint and reset the direction of scan. When multiple navigable viewpoints are present, the agent uniformly samples one.

In the second stage, an agent in HabitatSim retraces the viewpoint path and movements of the first stage generated by MatterSim, collecting mesh-rendered RGB and instance segmentation sensor data. The MatterSim RGB and HabitatSim RGB images are then aligned via FLANN-based feature matching [40], resulting in an alignment matrix that is used to place the MatterSim RGB and HabitatSim instance segmentation maps into alignment. The sequence of these MatterSim RGB and HabitatSim instance segmentation maps constitute an episode.

We keep objects of the following categories: picture, chair, lighting, cushion, table, plant, chest of drawers, towel, sofa, bed, appliances, stool, tv monitor, clothes, toilet, fireplace, furniture, bathtub, gym equipment.
Table 6: Continual & few-shot learning datasets

| Images          | Sequences | Classes | Content                  |
|-----------------|-----------|---------|--------------------------|
| Permuted MNIST  | 60K       | -       | -                        |
| Omniglot        | 32.4K     | -       | 1.6K Hand written characters |
| CIFAR-100       | 50K       | -       | 100 Common objects       |
| mini-ImageNet   | 50K       | -       | 100 Common objects       |
| tiered-ImageNet | 779K      | -       | 608 Common objects       |
| OpenLORIS       | 98K       | -       | 69 Small table-top obj.  |
| CORe50          | 164.8K    | 11      | 50 Hand-held obj.        |
| RoamingRooms (Ours) | 1.22M   | 6.9K    | 7.0K General indoor instances |

We initially generate 600 frames per sequence and remove all the frames with no object. Then we store every 200 image frames into a separate file.

During training and evaluation, each video sequence is loaded, and for each image we go through each object present in the image. We create the attention map using the segmentation groundtruth of the selected object. The attention map and the image together form a frame in our model input. For training, we randomly crop 100 frames from the sequence, and for evaluation we use the first 100 frames for deterministic results.

Please visit our released code repository to download the RoamingRooms dataset.

A.4 Semi-supervised Labels:

Here we describe how we sample the labeled vs. unlabeled flag for each example in the semi-supervised sequences in both RoamingOmniglot and RoamingRooms datasets. Due to the imbalance in our class distribution (from both the Chinese restaurant process and real data collection), directly masking the label may bias the model to ignore the rare seen classes. Ideally, we would like to preserve at least one labeled example for each class. Therefore, we designed the following procedure.

First, for each class $k$, suppose $m_k$ is the number of examples in the sequence that belong to the class. Let $\alpha$ be the target label ratio. Then the class-specific label ratio $\alpha_k$ is:

$$\alpha_k = (1 - \alpha) \exp(-0.5(m_k - 1)) + \alpha. \quad (10)$$

We then for each class $k$, we sample a binary Bernoulli sequence based on $\text{Ber}(\alpha_k)$. If a class has all zeros in the Bernoulli sequence, we flip the flag of one of the instances to 1 to make sure there is at least one labeled instance for each class. For all experiments, we set $\alpha = 0.3$.

A.5 Dataset Splits

We include details about our dataset splits in Table 7 and 8.

B Experiment Details

B.1 Network Architecture

For the RoamingOmniglot experiment we used the common 4-layer CNN for few-shot learning with 64 channels in each layer, resulting in a 64-d feature vector. For the RoamingRooms experiment we resize the input to 120×160 and we use the ResNet-12 architecture with 32,64,128,256 channels per block. To represent the feature of the input image with an attention mask, we concatenate the global average pooled feature with the attention ROI feature, resulting in a 512d feature vector. For the contextual RNN, in both experiments we used an LSTM with a 256d hidden state.

We use a linear layer to map from the output of the RNN to the features and control variables. We obtain $\gamma^{r,w}$ by adding 1.0 to the linear layer output and then applying the softplus activation. The bias units for $\beta^{r,w}$ are initialized to 10.0. We also apply the softplus activation to $m$ from the linear layer output.
Figure 7: Additional statistics about our RoamingRooms dataset.

Table 7: **Split information for RoamingOmniglot**. Each column is an alphabet and we include all the characters in the alphabet in the split. Rows are continuation of lines.

| Train          | Val         | Test         |
|----------------|-------------|--------------|
| Angelic        | Hebrew      | Old Church Slavonic |
| Grantha        | Early Aramaic| Slavonic    |
| N Ko           | Mkhedruli   | Gaelic       |
| Aurek-Besh     | Kannada     | Old Church Slavonic |
| Japanese (hiragana) | Manipuri  | Slavonic    |
| Malay          | Manipuri    | Greek       |
| Sanskrit       | Mongolian   | Latin       |
| Ojibwe         | Japanese (katakana) | Latin     |
| Korean         | Syriac (Serto) | Latin     |
| Arcadian       | Syriac (Serto) | Latin     |
| Greek          | Syriac (Serto) | Latin     |
| Alphabet of the Magi | Syriac (Serto) | Latin     |
| Blackfoot      | Tifinagh    | Latin       |
| Futurama       | Inuktitut   | Latin       |

**B.2 Training Procedure**

We use the Adam optimizer [25] for all of our experiments, with a gradient cap of 5.0. For RoamingOmniglot we train the network for 40k steps with a batch size 32 and maximum sequence length 150 across 2 GPUs and an initial learning rate 2e-3 decayed by 0.1 × at 20k and 30k steps. For RoamingRooms we train for 20k steps with a batch size 8 and maximum sequence length 100 across 4 GPUs and an initial learning rate 1e-3 decayed by 0.1 × at 8k and 16k steps. We use the BCE coefficient $\lambda = 1$ for all experiments. In semi-supervised experiments, around 30% examples are labeled when the number of examples grows large ($\alpha = 0.3$, see Equation 10). Early stopping is used in RoamingRooms experiments where the checkpoint with the highest validation AP score is chosen. For RoamingRooms, we sample Bernoulli sequences on unlabeled inputs to gradually allow semi-supervised writing to the prototype memory and we find it helps training stability. The probability starts with 0.0 and increase by 0.2 every 2000 training steps until reaching 1.0.

**B.3 Data Augmentation**

For RoamingOmniglot, we pad the 28×28 image to 32×32 and then apply random cropping.
Table 8: Split information for RoamingRooms. Each column is the ID of an indoor world. Rows are continuation of the lines.

|       | Split Worlds | Sequences | Frames   |
|-------|--------------|-----------|----------|
| Train | r1Q1z4bCv1o  | JmbYfDe2QKZ | 29hnd4uzFmX | ULaKaCPVFJRX E9uDoFAP3SH |
|       | 8WUmLavc2A   | Uxmj2M2itWa | mJXqFtmKg4 | V2XXFyvX4AS4d EUD6Fq7Sy2V |
|       | gYvKGZ5erqB  | gxdqLoLR6ruA | YFuZgdQ5vWj | GTV8GFCVJ9C sT4r6f7aTbPf |
|       | VYfe2KiqLan  | fzyznW3qQPFV | WYY71Yvysf5p8 | VfuaQ6m2Qom YmJxkBEshHNH |
|       | 2t7WJuJek07  | pLe4sQ7qrG  | cV4RvZv5u5T | XcA2TjSSAJ ur6pFq6Qu1A |
|       | 1pxnUDYAj8r  | b8cTzDMGq8G | x8B5xyUw9yE | X7HyMhZNos0 aayBhsNa07d |
|       | TbHJrujSAJp  | sKLMLpTheUy | 2axz1b91icZz | 2n8kARJN3HM Vfot9Ly1tCj |
|       | S9hNv5q7aGM  | EDJbREhgzL | qoiz87EJWz2 | 9v5SoiVnciC Vt2qJdWjCF2 |
|       | VzqgbhrpDEA  | D7G3Y4VRnH  | ZMojNkEp431 | uNh9QRRL6Hy 5LpH3gDmAk7 |
|       | rqfALeo17q   | e9zR4mvMw7  | yqstmuAEvh | zsnO4BB9uLZ JF19KD82E Mey |
|       | 759x9y6JkW5  | wc2JMjhGNoZ | BnPC6Dw41MGE | jh4fC6c5q5qQ HxppQynjfsfn |
|       | GdvgFv5R1Z5  | keEZ7cmS4wCh | yvrNrrzPKCB | D7N2ECKX4sJ PX4nDJXHeRG |
| Val   | s8pmcisQ38h  | dhjEzFouFzH | RFpmz2sNmrY 1LxTfKjw3qL 8194nk5LbLH |
|       | jtcx69c1yFQV | QCtXc6B85sX | p5wJjkQbkXX | JeGF25nY1j2p 82sE5b5LpXEH |
| Test  | oLBMNvg9in8  | r47D6H71a5e | ZMFMQCVibwU | VYUC4DcTcy RbAh3prwrgk9 |
|       | SN833Jr3Y2v  | g267yhEYnG | ac262MsG7at | 7y3aBxLe3Va B6lyBgMemKs |
|       | UvV83HsGw3v3 | VLzg9D0317F | 17DRP56b8fy | p4ot6Mv9nkk 52K3TnW8Zo |
|       | PukKg4mmmafe | Pm6F5ky3z2  | iSoynDfURQK | ARNzJeq3xb 5q7pVuzziYa |

For RoamingRooms, we apply random cropping in the time dimension to get a chunk of 100 frames per input example. We also apply random dropping of 5% of the frames. We pad the 120 × 160 images to 126 × 168 and apply random cropping in each frame. We also randomly flip the order of the sequence (going forward or backward).

### B.4 Spatiotemporal context experiment details

We use the Kylberg texture dataset [29] without rotations. Texture classes are split into train, val, and test, defined in Table 9. We resize all images first to 256 × 256. For each Omniglot image, a 28 × 28 patch is randomly cropped from a texture image to serve as background. Random Gaussian noises with mean zero and standard deviation 0.1 are added to the background images.

For spatial background experiments, we added an additional learnable network of the same size as the main network to take the background image as input, and output the same sized embedding vector. This embedding vector is further concatenated with the main embedding vector to form the final embedding of the input. We also found that using spatially overlaid images with a single CNN can achieve similar performance as well. The final numbers are reported using the concatenation approach since it is less prone to overlay noises and is more similar to the implementation we use in the RoamingRooms experiments.

### B.5 Baseline implementation details

**Online meta-learning (OML):** The OML model performs one gradient descent step for each input. In order for the model to predict unknown, we use the probability output from the softmax layer summing across the unused units. For example, if the softmax layer has 40 units and we have only seen 5 classes so far, then we sum the probability from the 6th to the last units. This summed probability is separately trained with a binary cross entropy, same as in Equation 8.

The inner learning rate is set to 1e-2 and we truncate the number of unrolled random descent steps to S/20 (RoamingOmniglot/RoamingRooms), in order to make the computation feasible. For
Table 9: Split information for the Kylberg texture dataset. Each column is an texture type. Rows are continuation of lines.

| Train           | blanket2 | ceiling2 | floor2 | grass1 | linseeds1 |
|-----------------|----------|----------|--------|--------|-----------|
|                 | pearlsugar1 | rice2    | scarf2 | screen1 | seat2     |
|                 | sesameseeds1 | stone1   | stoneslab1 |        |           |

| Val             | blanket1 | canvas1  | ceiling1 | floor1 | scarf1    |
|                 | rice1    | stone2   |          |        |           |

| Test            | wall1    | lentils1 | cushion1 | rug1   | sand1     |
|                 | oatmeal1 | stone3   | seat1    |        |           |

RoamingOmniglot, the network is trained with a batch size 32 across 2 GPUs, for a total of 20k steps, with an initial learning rate 2e-3 decayed by 0.1 at 10k and 16.7k steps. For RoamingRooms, the network is trained with a batch size 8 across 4 GPUs, for a total of 16k steps, with an initial learning rate 1e-3 decayed by 0.1 at 6.4k and 12.8k steps.

Long short-term memory (LSTM): We apply a stacked two layer LSTM with 256 hidden dimensions. Inputs are $h^{CNN}_t$ concatenated with the label one-hot vector. If an example is unlabeled, then the label vector is all-zero. We directly apply a linear layer on top of the LSTM to map the LSTM memory output into classification logits, and the last logit is the binary classification logit reserved for unknown. The training procedure is the same as our CPM model.

Differentiable neural computer (DNC): In order to make the DNC model work properly, we found that it is sometimes helpful to pretrain the CNN weights. Simply initializing from scratch and train CNN+DNC end-to-end sometimes results in poor performance. We hypothesize that the attention structure in the DNC model is detrimental to representation learning. Therefore, for RoamingOmniglot experiments, we use pretrained ProtoNet weights for solving 1-shot 5-way episodes to initialize the CNN, and we keep finetuning the CNN weights with 10% of the full learning rate. For RoamingRooms experiments, we train the full model end-to-end from scratch.

The DNC is also modified so that it is more effective using the label information from the input. In the original MANN paper [46] for one-shot learning, the input features $h^{CNN}_t$ and the label one-hot ID are simply concatenated to feed into the LSTM controller of MANN. We find that it is beneficial to directly add label one-hot vector as an input to the write head that generates the write attention and the write content. Similar to the LSTM model, the memory readout is also sent to a linear layer in order to get the final classification logits, and the last logit is the binary classification logit reserved for the unknowns. Finally we remove the linkage prediction part of the DNC due to training instability.

The controller LSTM has 256 hidden dimensions, and the memory has 64 slots each with 64 dimensions. There are 4 read heads and 4 write heads. The training procedure is the same as CPM.

Online ProtoNet: Online ProtoNet is our modification of the original ProtoNet [50]. It is similar to our CPM model without the contextual RNN. The feature from the CNN is directly written to the prototype memory. In addition, we do not predict the control hyperparameters $\beta_t^{\{r,w\}}, \gamma_t^{\{r,w\}}$ from the RNN and they are learned as regular parameters. The training procedure is the same as CPM.

Online MatchingNet: Online MatchingNet is our modification of the original MatchingNet [53]. We do not consider the context embedding in the MatchingNet paper since it was originally designed for the entire episode using an attentional RNN encoder. It is similar to online ProtoNet but instead of doing online averaging, it directly stores each example and its class. Since it is an example-based storage, we did not extend it to learn from unlabeled examples, and all unlabeled examples are skipped. We use a similar decision rule to determine whether an example belongs to a known cluster by looking at the distance to the nearest exemplar stored in the memory, shifted by $\beta$ and scaled by $1/\gamma$. Note that online MatchingNet is not efficient at memory storage since it scales with the number of steps in the sequence. In addition, we use the negative Euclidean distance as the similarity function. The training procedure is the same as CPM.
Online infinite mixture prototypes (IMP): Online IMP is proposed as a mix of prototype and example-based storage by allowing a class to have multiple clusters. If an example is classified as unknown or it is unlabeled, we will assign its cluster based on our prediction, which either assigns it to one of the existing clusters or creates a new cluster, depending on its distance to the nearest cluster. If a cluster with an unknown label later is assigned with an example with a known class, then the cluster label will also be updated. We use the same decision rule as online ProtoNet to determine whether an example belongs to a known cluster by looking at the distance to the nearest cluster, shifted by $\beta$ and scaled by $1/\gamma$. As described above, online IMP has the capability of learning from unlabeled examples, unlike online MatchingNet. However similar to online MatchingNet, online IMP is also not efficient at memory storage since in the worst case it also scales with the number of steps in the sequence. Again, the training procedure is the same as CPM.

C Additional Visualization of Experimental Results

C.1 Video Visualization

We include video visualization of RoamingRooms sequences here: [Video Link 1](https://drive.google.com/drive/folders/1gBJBFdWboOE0vK6CEYKxIL10gO_jrqr8). Our CPM model prediction can be found here: [Video Link 2](https://drive.google.com/drive/folders/1rp9xXAcrrZyffngPdtoS9B16P9uxJ9xK)

C.2 Embedding Visualization

Figure 8 shows the learned embedding of each example in Online ProtoNet vs. our CPM model in RoamingOmniglot sequences, where colors indicate environment IDs. In Online ProtoNet, the example features do not reflect the temporal context, and as a result, colors are scattered across the space. By contrast, in the CPM embedding visualization, colors are clustered together and we see a smoother transition of environments in the embedding space.

Figure 8: **Embedding space visualization of RoamingOmniglot sequences using t-SNE [37]**. Different color denotes different environments. Text labels (relative to each environment) are annotated beside the scatter points. Unlabeled examples shown in smaller circles with lighter colors. **Left**: Online ProtoNet; **Right**: CPM. The embeddings learned CPM model shows a smoother transition of classes based on their temporal environments.
Figure 9: CPM control parameters \((\beta_r^w, \gamma_r^w)\) vs. time. Left: RoamingOmniglot sequences; Right: RoamingRooms sequences; Top: \(\beta_r^w\) the threshold parameter; Bottom: \(\gamma_r^w\) the temperature parameter.

C.3 Control Parameters vs. Time

Finally we visualize the control parameter values predicted by the RNN in Figure 9. We verify that we indeed need two sets of \(\beta\) and \(\gamma\) for read and write operations separately as they learn different values. \(\beta_w\) is smaller than \(\beta_r\) which means that the network is more conservative when writing to prototypes. \(\gamma_w\) grows larger over time, which means that the network prefers a softer slope when writing to prototypes since in the later stage the prototype memory has already stored enough content and it can grow faster, whereas in the earlier stage, the prototype memory is more conservative to avoid embedding vectors to be assigned to wrong clusters.