On Sequential Bayesian Inference for Continual Learning

Samuel Kessler [1,*], Adam Cobb [3], Tim G. J. Rudner [2], Stefan Zohren [1] and Stephen J. Roberts [1]

Table of Contents

Table of Contents

Continual Learning

In machine learning, most of the time we develop models assuming:

- a fixed distribution over the data
- a fixed objective function

Continual learning (CL) considers the case where the data-distribution and/or objective function can change over time.

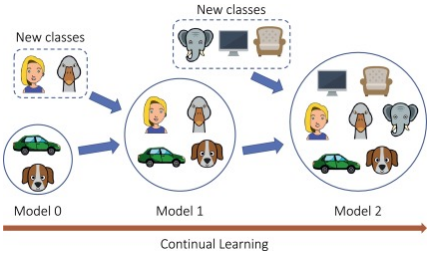


Figure: Class incremental CL (Michieli, Toldo, and Zanuttigh 2022).

Continual Learning: Tasks

- We split a CL setting into a series of consecutive tasks

$$\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \ldots$$

  where we believe each task has a fixed distribution over the data and a fixed objective function.
- i.e. task $\mathcal{T}_t$ is comprised of a dataset $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^{N_t}$
- The challenge is to learn sequentially on each task **without forgetting what was learnt in previous tasks**.

Continual Learning: Cases

Lesort et al. 2019 provide some good examples where CL is necessary:

- You want to update a trained model but no longer have access to the original training data.
- You want to train a model on a sequence of tasks but can't use the whole dataset at once (due to memory or computational constraints).
- You want an agent that learns multiple policies but you don't know when/how the learning objective changes.
- You want to learn from a continuous stream of data that changes through time (and perhaps you don't know when/how).

Continual Learning: Examples

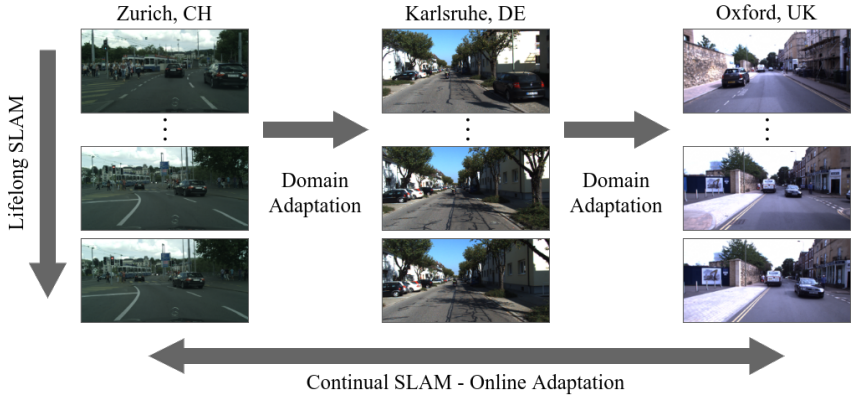


Figure: Continual SLAM (Vödisch et al. 2023).

Continual Learning: Examples

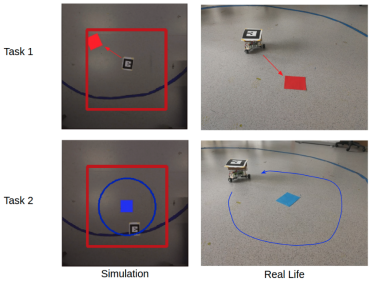


Figure 1. Having access to task 1 only first, and then task 2 only, we learn a single policy that solves two real-life navigation tasks using policy distillation and sim2real transfer.

Figure: Sequential Objectives (Traoré et al. 2019).

Continual Learning: Examples

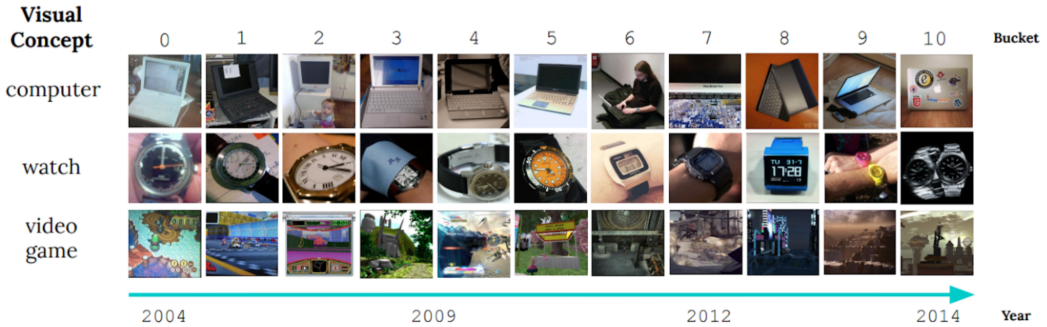


Figure: Conceptual Drift (Jain and Shenoy 2023).

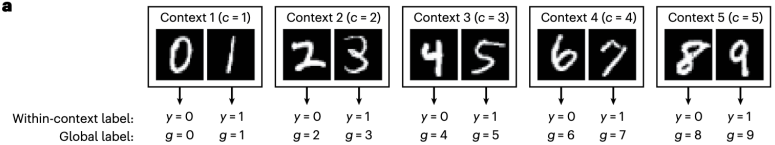
Continual Learning Settings

- **Task-incremental**: The model is allowed to know which task the test data comes from (so output space is task-dependent).
- **Domain-incremental**: The model doesn't know which task the test data comes from, and each task uses an equivalent but separate output space.
- **Class-incremental**: The model doesn't know which task the test data comes from, and the output space is shared across tasks.

Continual Learning: Split-MNIST

MNIST but each task only contains training points from two classes.

a



b

	Input (at test time)	Expected output	Intuitive description
Task-incremental learning	Image + context label	Within-context label[a]	Choice between two digits of same context (e.g. 0 or 1)
Domain-incremental learning	Image	Within-context label	Is the digit odd or even?
Class-incremental learning	Image	Global label	Choice between all ten digits

Figure: Split MNIST (Ven, Tuytelaars, and Tolias 2022).

Continual Learning Settings

- **Single-headed**: Use the same model for all tasks.
- **Multi-headed**: Use a base model f for all tasks, but train separate heads g_t for each task and use some composition/combination of the heads at test time.
  - (Or just pick the correct head if you know which task the test data comes from.)

Table of Contents

Sequential Bayesian Inference

- At $t = 1$ (for task $\mathcal{T}_1$) the posterior predictive for test point $x_1^* \in \mathcal{D}_1$:

$$p(y_1^*|x_1^*, \mathcal{D}_1) = \int p(y_1^*|x_1^*, \theta)p(\theta|\mathcal{D}_1)d\theta$$

- At $t = 2$ (for task $\mathcal{T}_2$) the posterior predictive for test point $x_2^* \in \mathcal{D}_1 \cup \mathcal{D}_2$:

$$p(y_2^*|x_2^*, \mathcal{D}_1, \mathcal{D}_2) = \int p(y_2^*|x_2^*, \theta)p(\theta|\mathcal{D}_1, \mathcal{D}_2)d\theta$$

- Each time step T we receive $\mathcal{D}_T$ and update the posterior over θ:

$$p(\theta|\mathcal{D}_1, \ldots, \mathcal{D}_{T-1}, \mathcal{D}_T) = \frac{p(\mathcal{D}_T|\theta)p(\theta|\mathcal{D}_1, \ldots, \mathcal{D}_{T-1})}{p(\mathcal{D}_T|\mathcal{D}_1, \ldots, \mathcal{D}_{T-1})}$$

Sequential Bayesian Inference

- Each time step T we receive $\mathcal{D}_T$ and update the posterior over θ:

$$p(\theta|\mathcal{D}_1, \ldots, \mathcal{D}_{T-1}, \mathcal{D}_T) = \frac{p(\mathcal{D}_T|\theta)p(\theta|\mathcal{D}_1, \ldots, \mathcal{D}_{T-1})}{p(\mathcal{D}_T|\mathcal{D}_1, \ldots, \mathcal{D}_{T-1})}$$

(assuming the datasets $\mathcal{D}_1, \ldots, \mathcal{D}_T$ are conditionally independent given θ).

  - $p(\mathcal{D}_T|\theta)$ doesn't require previous datasets.
  - $p(\theta|\mathcal{D}_1, \ldots, \mathcal{D}_{T-1})$ is the posterior of time step $T-1$ and can be seen as the prior for time step T.
  - $p(\mathcal{D}_T|\mathcal{D}_1, \ldots, \mathcal{D}_{T-1})$ is a constant we can ignore (sort of).

This Paper's Thesis

Kessler et al. 2023:

- Sequential Bayesian inference *should* be well suited for CL...
- But over **the weights of** a BNN, sequential Bayes isn't actually very good in practice.
  1. Even if a BNN is a well-specified/flexible-enough model for the posterior.
  2. Even if we are able to perform high-quality inference.
  3. In part due to class imbalance across tasks.
- However, if we do sequential Bayes on the model in some lower-dimension latent space, might we get better results?

Table of Contents

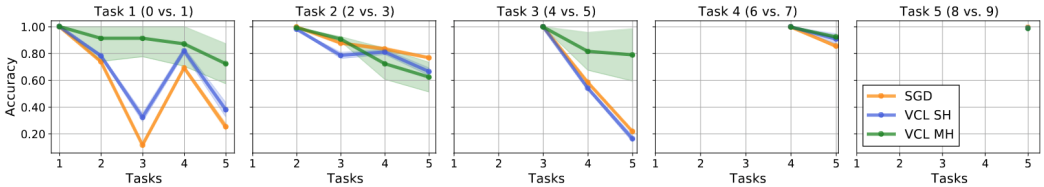
Variational Continual Learning (VCL)

One generic solution: optimise an approximate posterior $q_t(\theta|\mathcal{D}_{1:t}) =: q_t(\theta)$ at each time step t via (Nguyen et al. 2018):

$$\mathcal{L}(\theta|\mathcal{D}_t) = \mathbb{D}_{KL}[q_t(\theta)||q_{t-1}(\theta|\mathcal{D}_{1:t-1})] - \mathbb{E}_{q_t}[\log p(\mathcal{D}_t|\theta)].$$

Use single- and multi-headed versions (SH and MH respectively).

VCL vs SGD (on a 2-layer BNN)

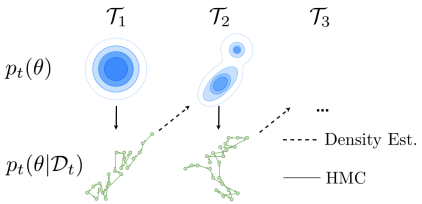


- VCL-MH is kind of cheating (and shouldn't really be necessary)
- What's worrying is that VCL-SH is pretty much the same as SGD

Hypothesis: the failure is due to imperfect approximate inference of the posterior(s).

Table of Contents

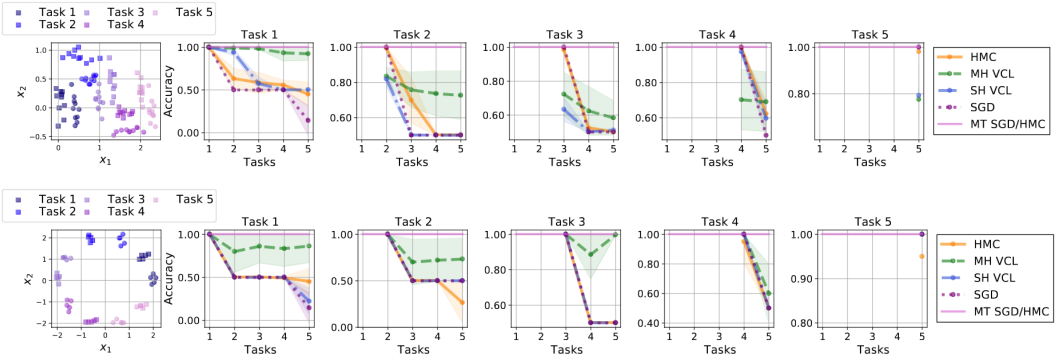
HMC



Let's try the gold standard for inference: HMC

- Start with prior
- Get posterior samples from task 1 via HMC
- Turn those samples into a GMM approx. posterior via by density estimation
- Repeat with this approx posterior as the new prior for task 2, &c.

Trying HMC for better inference



- HMC is only marginally better than SGD.

Table of Contents

Why doesn't it work?

Two potential reasons:

1. Model misspecification
2. Imperfect inference

Model Misspecification

- Consider the simple model where $\theta_1, \theta_2, \ldots, \theta_t \sim p$ and observations arrive online (with known σ):

$$p(y_t|\theta_t) = \mathcal{N}(y_t; \theta_t, \sigma^2)$$

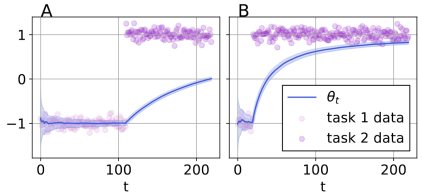
- We want to infer the filtering distribution $p(\theta_t|y_1, \ldots, y_t)$.
- We can perform exact sequential Bayesian inference via closed-form updates:

$$\hat{\theta}_t = \hat{\sigma}_t^2 \left(\frac{y_t}{\sigma^2} + \frac{\hat{\theta}_{t-1}}{\hat{\sigma}_{t-1}^2} \right) = \hat{\sigma}_t^2 \left(\sum_{i=1}^{t} \frac{y_i}{\sigma^2} + \frac{\hat{\theta}_0}{\hat{\sigma}_0^2} \right), \quad \frac{1}{\hat{\sigma}_t^2} = \frac{1}{\sigma^2} + \frac{1}{\hat{\sigma_{t-1}^2}}$$

Given some prior $p(\hat{\theta}_0) = \mathcal{N}(\theta_0; 0, \sigma_0^2)$.

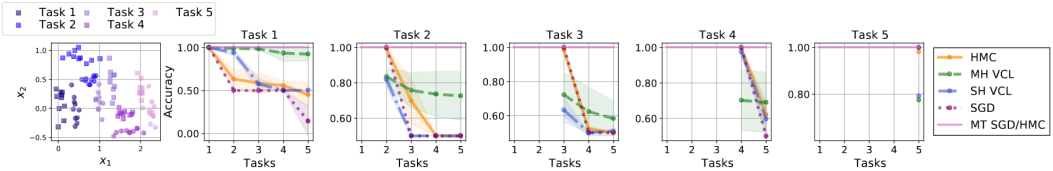
Model Misspecification

- Try to model changepoint detection where data moves from a $\mathcal{N}(-1, \sigma^2)$ distribution to $\mathcal{N}(1, \sigma^2)$ after some number of time steps (θ moves from -1 to 1).



- A linear model doesn't work when subsequent tasks have different modes (kind of obvious). It will track the global mean.
- "*Despite performing exact inference, a misspecified model can forget*"

Model Misspecification



- Note that in the previous experiment we don't think that the BNN is misspecified: when trained on all tasks at once the BNN is able to get 100% accuracy (pink line).
- So it must be a problem with the way sequential Bayesian inference works on a BNN.

Imbalanced Task Data

- The authors analyse a simplified model of sequential Bayes on a BNN.

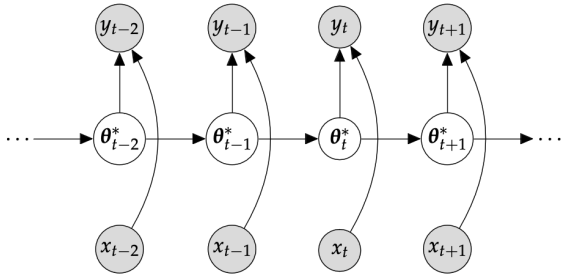


Figure A8. Graphical model of under which we perform inference in Section 5. Grey nodes are observed and white are latent variables.

Imbalanced Task Data

- They argue that, just like in the previous experiment, uneven amounts of data per task can also affect the posterior obtained as time goes on.

Lemma 1. *Under Assumptions 1 and 2 the dynamics and likelihood are Gaussian. Thus, we are able to infer the posterior distribution over the optimal weights using Bayesian updates and by linearizing the BNN the update equations for the posterior of the mean and variance of the BNN for a new data point are:*

$$\mu_{t,post} = \sigma^2_{t,post}\left(\frac{\mu_{t,prior}}{\sigma^2_{t,prior}(\eta^2)} + \frac{y_t}{\sigma^2} g(\boldsymbol{x}_t)\right) \qquad \text{and} \qquad \frac{1}{\sigma^2_{t,post}} = \frac{g(\boldsymbol{x}_t)^2}{\sigma^2} + \frac{1}{\sigma^2_{t,prior}(\eta^2)}, \quad (12)$$

where we drop the notation for the i-th parameter, the posterior is $\mathcal{N}(\theta^*_t; \mu_{t,post}, \sigma^2_{t,post})$ *and* $g(\boldsymbol{x}_t) = \frac{\partial f(\boldsymbol{x}_t, \theta^*_{it})}{\partial \theta^*_{it}}$ *and* $\sigma^2_{t,prior}$ *is a function of* η^2.

Imbalanced Task Data

- They argue that, just like in the previous experiment, uneven amounts of data per task can also affect the posterior obtained as time goes on.

Lemma 1. *Under Assumptions 1 and 2 the dynamics and likelihood are Gaussian. Thus, we are able to infer the posterior distribution over the optimal weights using Bayesian updates and by linearizing the BNN the update equations for the posterior of the mean and variance of the BNN for a new data point are:*

$$\mu_{t,post} = \sigma^2_{t,post}\left(\frac{\mu_{t,prior}}{\sigma^2_{t,prior}(\eta^2)} + \frac{y_t}{\sigma^2} g(\boldsymbol{x}_t)\right) \quad \text{and} \quad \frac{1}{\sigma^2_{t,post}} = \frac{g(\boldsymbol{x}_t)^2}{\sigma^2} + \frac{1}{\sigma^2_{t,prior}(\eta^2)}, \tag{12}$$

where we drop the notation for the i-th parameter, the posterior is $\mathcal{N}(\theta_i^*; \mu_{t,post}, \sigma^2_{t,post})$ *and* $g(\boldsymbol{x}_t) = \frac{\partial f(\boldsymbol{x}_t;\theta_{it}^*)}{\partial \theta_{it}^*}$ *and* $\sigma^2_{t,prior}$ *is a function of* η^2.

- "*the posterior mean depends linearly on the prior and a data-dependent term and so will behave similarly to our previous example*".

Imbalanced Task Data

- This is a problem for two main reasons:
  1. In most online settings we might not know how long each task lasts for, i.e. how much data to expect per task.
  2. (Task-incremental) CL algorithms almost always require the use of *core sets* of data from previous tasks.
     - These saved previous datapoints are sprinkled into subsequent tasks to prevent catastrophic forgetting.

Table of Contents

Prototypical Bayesian Continual Learning

- Problem: performing sequential Bayesian inference over BNN weights is probably a bad idea.
- Solution: instead, do it over latent variables in an embedding space (where the embeddings come from a regular NN).

ProtoCL Model

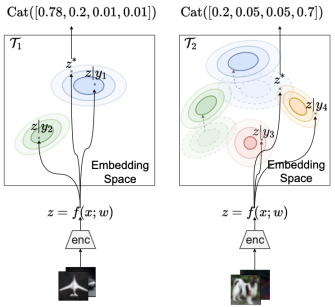


Figure 6. Overview of ProtoCL.

- Use regular NN to embed data points $z = f(x; w)$ with parameters w.
- Also keep track of Gaussian 'prototype' distributions per class y in the embedding space at time t.

$$\bar{z}_{yt} \sim \mathcal{N}(\mu_{yt}, \Lambda_{yt}^{-1})$$

- We'll then use MAP estimates to assign class (based on which class prototype the embedding z is closest to).

ProtoCL Model

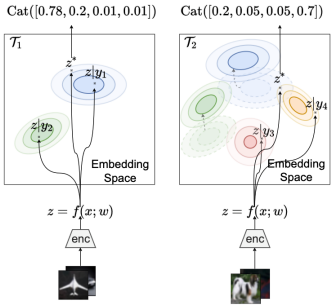


Figure 6. Overview of ProtoCL.

- For data point i of task t, model $x_{i,t}$'s class $y_{i,t}$ via a categorical distribution with a Dirichlet prior:

$$y_{i,t} \sim \mathrm{Cat}(p_{1:J}), \quad p_{1:J} \sim \mathrm{Dir}(\alpha_t)$$

- Given a class $y_{i,t}$ we assume the embedding z_{it} is distributed normally around the class' prototype $\bar{z}_{yt}$.

$$z_{it}|y_{i,t} \sim \mathcal{N}(\bar{z}_{yt}, \Sigma_\epsilon), \quad \bar{z}_{yt} \sim \mathcal{N}(\mu_{yt}, \Lambda_{yt}^{-1})$$

ProtoCL Model

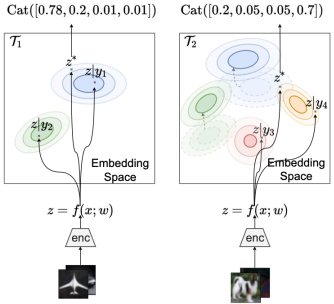


Figure 6. Overview of ProtoCL.

- The posterior distribution over
  - class probabilities $\{p_j\}_{j=1}^J$ and
  - class embeddings $\{\bar{z}_{y_j}\}_{j=1}^J$ (prototypes)

  is denoted by $p(\theta)$ with parameters

$$\eta_t = \{\alpha_t, \mu_{1:J,t}, \Lambda_{1:J,t}^{-1}\}$$

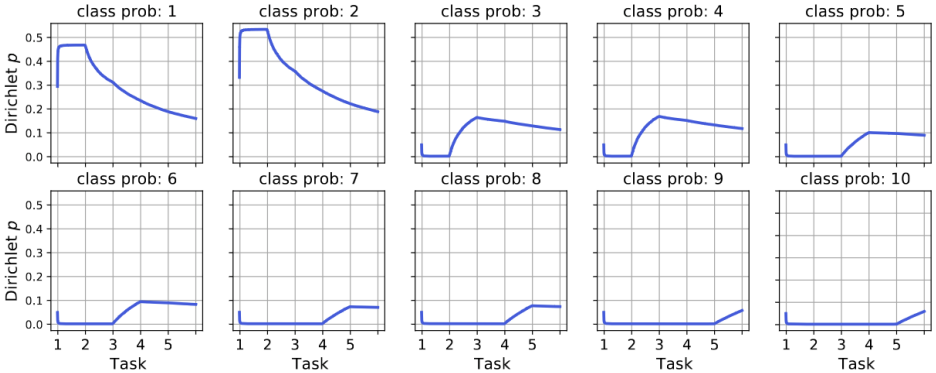
ProtoCL Updates

$$\alpha_{t+1,j} = \alpha_{t,j} + \sum_{i=1}^{N_t} \mathbb{I}(y_t^i = j),$$

where N_t is the number of points seen during the update at timestep t.

ProtoCL Update

(Update of α_t as tasks progress in Split MNIST.)



ProtoCL Updates

$$\alpha_{t+1,j} = \alpha_{t,j} + \sum_{i=1}^{N_t} \mathbb{I}(y_t^i = j),$$

where N_t is the number of points seen during the update at timestep t.

$$\Lambda_{y_{t+1}} = \Lambda_{y_t} + N_y \Lambda_\epsilon^{-1}$$
$$\mu_{y_{t+1}} = \Lambda_{y_{t+1}}^{-1}(\Lambda_{y_t}\mu_{y_t} + N_y \Lambda_\epsilon^{-1} \bar{z}_{y_t}),\ \forall y_t \in C_t$$

where N_y is the number of samples of class y and $\bar{z}_{y_t} = (1/N_y)\sum_{i=1}^{N_y} z_{yi}$.

ProtoCL Objective

- Posterior predictive distribution of the embeddings z and class labels y

$$p(z, y) = p(y) \prod_{i=1}^{N_t} \mathcal{N}(z_{it} | y_{it}; \mu_{y_t,t}, \Sigma_\epsilon + \Lambda_{y_t,t}^{-1})$$

  where $p(y) = \alpha_y / \sum_{j=1}^{J} \alpha_j$
- Optimize this objective with gradient-based optimization to learn embedding $z = f(x; w)$.

ProtoCL Algorithm

Algorithm 1 ProtoCL continual learning

Input: task datasets $\mathcal{T}_{1:T}$, initialize embedding function: $f(\cdot\,; \boldsymbol{w})$, coreset: $\mathcal{M} = \emptyset$.
for $\mathcal{T}_1$ **to** $\mathcal{T}_T$ **do**
  for each batch in $\mathcal{T}_i \cup \mathcal{M}$ **do**
    Optimize $f(\cdot; \boldsymbol{w})$ by maximizing the posterior predictive $p(\mathbf{z}, y)$ Equation (18)
    Obtain posterior over $\boldsymbol{\theta}$ by updating η, Equations (15)–(17).
  end for
  Add random subset from $\mathcal{T}_i$ to $\mathcal{M}$.
end for

ProtoCL Results

Method	Coreset	Split-MNIST	Split-FMNIST
VCL [9]	✗	33.01 ± 0.08	32.77 ± 1.25
+ coreset	✓	52.98 ± 18.56	61.12 ± 16.96
HIBNN * [11]	✗	85.50 ± 3.20	43.70 ± 20.21
FROMP [22]	✓	84.40 ± 0.00	68.54 ± 0.00
S-FSVI [47]	✓	$\mathbf{92.94 \pm 0.17}$	80.55 ± 0.41
ProtoCL (ours)	✓	$\mathbf{93.73 \pm 1.05}$	$\mathbf{82.73 \pm 1.70}$

ProtoCL Results

Method	Training Time (s) (↓)	Split CIFAR-10 (Acc) (↑)
FROMP [22]	1425 ± 28	48.92 ± 10.86
S-FSVI [47]	44434 ± 91	50.85 ± 3.87
ProtoCL (ours)	$\mathbf{384 \pm 6}$	$\mathbf{55.81 \pm 2.10}$
		Split CIFAR-100 (Acc) (↑)
S-FSVI [47]	37355 ± 1135	20.04 ± 2.37
ProtoCL (ours)	$\mathbf{1425 \pm 28}$	$\mathbf{23.96 \pm 1.34}$

ProtoCL: Criticisms

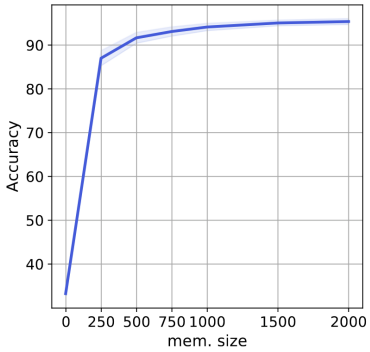


Figure A6. Split-MNIST average test accuracy over five tasks for different memory sizes. On the x-axis, we show the size of the entire memory buffer shared by all five tasks. Accuracies are over a mean and standard deviation over five different runs with different random seeds.

- Core set
- Heavily dependent on embedding dimension
  - (F)MNIST: 128
  - CIFAR10/100: 32

ProtoCL: Criticisms

"*The stated aim of ProtoCL is not to provide a novel state-of-the-art method for CL, but rather to propose a simple baseline that takes an alternative route than weight-space sequential Bayesian inference.*"

References I

Jain, Nishant and Pradeep Shenoy (Dec. 2023). *Instance-Conditional Timescales of Decay for Non-Stationary Learning*. en. arXiv:2212.05908 [cs]. URL: http://arxiv.org/abs/2212.05908 (visited on 02/19/2024).

Kessler, Samuel et al. (May 2023). "On Sequential Bayesian Inference for Continual Learning". en. In: *Entropy* 25.6. arXiv:2301.01828 [cs], p. 884. ISSN: 1099-4300. DOI: 10.3390/e25060884. URL: http://arxiv.org/abs/2301.01828 (visited on 02/16/2024).

Lesort, Timothée et al. (Nov. 2019). *Continual Learning for Robotics: Definition, Framework, Learning Strategies, Opportunities and Challenges*. en. arXiv:1907.00182 [cs]. URL: http://arxiv.org/abs/1907.00182 (visited on 02/19/2024).

References II

Michieli, Umberto, Marco Toldo, and Pietro Zanuttigh (Jan. 2022). "Chapter 8 - Domain adaptation and continual learning in semantic segmentation". In: *Advanced Methods and Deep Learning in Computer Vision*. Ed. by E. R. Davies and Matthew A. Turk. Computer Vision and Pattern Recognition. Academic Press, pp. 275–303. ISBN: 978-0-12-822109-9. DOI: 10.1016/B978-0-12-822109-9.00017-5. URL: https://www.sciencedirect.com/science/article/pii/B9780128221099000175 (visited on 02/19/2024).

Nguyen, Cuong V. et al. (May 2018). *Variational Continual Learning*. en. arXiv:1710.10628 [cs, stat]. URL: http://arxiv.org/abs/1710.10628 (visited on 02/16/2024).

References III

Traoré, René et al. (June 2019). *Continual Reinforcement Learning deployed in Real-life using Policy Distillation and Sim2Real Transfer*. en. arXiv:1906.04452 [cs, stat]. URL: http://arxiv.org/abs/1906.04452 (visited on 02/19/2024).

Ven, Gido van de, Tinne Tuytelaars, and Andreas Tolias (Dec. 2022). "Three types of incremental learning". In: *Nature Machine Intelligence* 4, pp. 1–13. DOI: 10.1038/s42256-022-00568-3.

Vödisch, Niclas et al. (2023). "Continual SLAM: Beyond Lifelong Simultaneous Localization and Mapping Through Continual Learning". In: vol. 27. arXiv:2203.01578 [cs], pp. 19–35. DOI: 10.1007/978-3-031-25555-7_3. URL: http://arxiv.org/abs/2203.01578 (visited on 02/19/2024).