Towards Robust Evaluations of Continual Learning

Sebastian Farquhar
Department of Computer Science
University of Oxford
sebastian.farquhar@balliol.ox.ac.uk

Yarin Gal
Department of Computer Science
University of Oxford
yarin@cs.ox.ac.uk

Abstract

Continual learning experiments used in current deep learning papers do not faithfully assess fundamental challenges of learning *continually*, masking weak-points of the suggested approaches instead. We study gaps in such existing evaluations, proposing essential experimental evaluations that are more representative of continual learning's challenges, and suggest a re-prioritization of research efforts in the field. We show that current approaches fail with our new evaluations and, to analyse these failures, we propose a variational loss which unifies many existing solutions to continual learning under a Bayesian framing, as either 'prior-focused' or 'likelihood-focused'. We show that while prior-focused approaches such as EWC and VCL perform well on existing evaluations, they perform dramatically worse when compared to likelihood-focused approaches on other simple tasks.

1 Introduction

Certain applications require *continual* learning—splitting training into a series of tasks and discarding the data after training each task. If data relate to individuals it may be unethical or illegal to keep old datasets: for example, a hospital might need to delete patient data after a certain point. In other applications, real-time industrial systems could face distributional shifts with an underlying distribution which changes faster than the time it would take to retrain a new model with all the data. Neural networks trained on such series of tasks tend to forget earlier tasks (often referred to as *catastrophic forgetting*). Recent works have shown promising advances towards continual learning by adding regularization terms to the loss function which preserve important parameters [14, 22, 26, 34].

However, experimental evaluations in these recent works have major blind-spots which mask weak-points of current approaches. These works make use of several lightweight experiments to indicate directional performance, including evaluations based on MNIST [15]. For example, such evaluations have blindspots because they allow networks not to change much between tasks, or, as we will see below, are set up in ways which do not reflect the core motivation for learning continually. Evaluations which obscure shortcomings of suggested continual learning solutions impede developments in the field, since new works are not aware of limitations of past research which need to be addressed.

In this paper we analyse the limitations of existing continual learning approaches and the blind-spots of current evaluations. For our analysis we develop a unifying Bayesian perspective which contrasts 'prior-focused' approaches (approaches which use regularization to treat old parameters as a prior) against 'likelihood-focused' approaches (approaches which estimate the likelihood of old data according to the current model, generally through a sort of memory). Through a likelihood-focused derivation of a variational inference (VI) loss, this unified view provides theoretical grounding for existing work such as Shin et al. [30]. It additionally grounds a novel approach we suggest, *Variational Generative Experience Replay* (VGER), which we introduce as the closest appropriate likelihood-focused alternative to Variational Continual Learning (VCL, the current state-of-the-art prior-focused approach [22]). The prior- and likelihood-focused approaches loosely correspond to regularization and memory-based approaches distinguished by other authors [24]. However, our Bayesian perspective,

Preprint. Work in progress.

rather than a neuroscientific one [27], draws out the difference between likelihood-focused approaches and other memory-based approaches that store experiences for reasons other than estimating past likelihoods (such as [18]). We then study the limitations of prior-focused approaches and demonstrate, for example, that approaches such as VCL rely on the network parameters not to change much between tasks, a limitation which likelihood-focused approaches alleviate.

At the same time, we analyse the blind-spots of current continual learning *evaluations*, offering both a theoretical analysis of the evaluations' shortcomings and empirical evidence to show their bias towards 'prior-focused' approaches. We show that existing lightweight evaluations in continual learning can be misleading, because models which perform well according to them can perform very badly on similar, but more representative, continual learning evaluations. To remedy this, we propose modifications to existing experimental evaluations to make them more representative of the challenges that motivate the field. We show that although prior- and likelihood-focused approaches perform similarly on the existing evaluations, likelihood-focused approaches perform much better on our more representative suggested tasks. The main contribution in this paper is the suggestion, and support, for a re-prioritization of research efforts in the field of continual learning, by analysing failure modes of current solutions and evaluations.

2 Existing Work

Approaches to continual learning and catastrophic forgetting form three main families. First, prior-focused approaches use regularization to create 'elastic' parameters. These include: Elastic Weight Consolidation (EWC) [14], Synaptic Intelligence (SI) [34], and Variational Continual Learning (VCL) [22]. VCL is a variational inference (VI) method that uses Bayesian neural networks [1, 19, 20]. VCL sets the posterior at the end of training a task to be the prior when beginning training for the next task. To improve performance, VCL can also use coresets—memorizing very small representative samples of past data. VCL with coresets fine-tunes on the withheld coreset data just before testing. EWC and SI instead use ordinary neural networks with a regularization term added to the loss which reflects a Gaussian prior for each parameter whose means are the old parameters. They use the approximate Fisher information as a way of estimating the Hessian to assess importance, which implicitly sets the variance of each parameter prior. Ritter et al. [26] extend this work with a richer estimate of importance using a block-diagonal estimate of the Hessian. Prior-focused approaches have received considerable attention in part because the methods are sufficiently general to apply to many architectures and settings.

Second, likelihood-focused approaches estimate the likelihood of data from old tasks under the current model, in addition to the likelihood of the data from the current task. This estimation might use a generative model as in Deep Generative Replay (DGR) by Shin et al. [30]. The coresets in VCL serve the role of helping estimate past likelihoods. The likelihood-focused family of approaches largely overlaps with the 'dual-memory learning systems' which build on Robins [27] following a neuroscientific motivation [24].

Last, dynamic architecture approaches involve changing the structure of the networks in significant ways to incorporate learning from each task separately (e.g., Li & Hoiem [17], Rusu et al. [28]). A more thorough survey of these as well as prior- and likelihood-focused approaches can be found in Appendix A. In this paper, we consider only prior- and likelihood-focussed approaches as these are more generally applicable to a wide range of settings and architectures.

2.1 Experimental Evaluations

In this paper we assess the robustness of common experimental evaluations in current continual learning literature. In this section we give an in-depth review of key experimental evaluations as they are used in *current research*. These are lightweight tests of continual learning performance, and of crucial importance are experiments using variations of the MNIST dataset: *Permuted MNIST* and *Split MNIST*, most commonly in *multi-headed* form. In the next sections §3 we will offer a critical analysis of these experimental evaluations, which will support our suggested alternative evaluations thereafter.

2.1.1 Permuted MNIST

The Permuted MNIST experiment was introduced by Goodfellow et al. [8]. In their experiment, a model is trained on MNIST as $\mathcal{D}_0$. Each later $\mathcal{D}_t$ for $1 \leq t < 10$ is constructed from the MNIST

data but with the pixels of each digit randomly permuted in the same way. After training on each dataset, one evaluates the model on each of the previous datasets as well as the current. Goodfellow et al. [8] used this experiment to investigate feature extraction, but it has since become a mainstay for continual learning evaluation [14, 16, 18, 22, 26, 30, 34].

2.1.2 Split MNIST

The Split MNIST experiment was introduced by Zenke et al. [34] in a *multi-headed* form and used by other authors including Shin et al. [30] and Nguyen et al. [22] (Ritter et al. [26] use a two-task variant). The experiment constructs a series of five related tasks. The first task is to distinguish the digits (0, 1), then (2, 3) etc. All papers use the multi-headed version although it could be *single-headed* or *multi-headed* variant (Permuted MNIST is always performed single-headed). In the single-headed variant the model always makes a prediction over the classes [0:9]. The multi-headed variant is much easier to solve. In the multi-headed variant the model prediction is constrained to be only from the two classes represented in each task. For example, when evaluating the performance on the first task, the model only needs to predict probabilities for zero versus one. In some cases, multi-heading is taken even further and training is only done on the head governing the specific classes included in the task [22, 26, 34]. As a result, training does not drive predicted probabilities for unseen heads to zero, as they would in the single-headed case. Multi-heading is often used in similar non-MNIST evaluations, for example, in Nguyen et al. [22], Ritter et al. [26].

2.1.3 Two-task transfer

In some works a two-task transfer learning evaluation is used for continual learning. For example, Shin et al. [30] train first on MNIST and then on SVHN [21] (or vice versa) in order to see whether their algorithm preserves performance on the first task after training on the second. Jung et al. [12] and Li & Hoiem [17] perform similar experiments.

3 Critical Analysis of Existing Evaluations

Permuted MNIST and multi-headed Split MNIST are commonly used to compare continual learning algorithms. However, each of these makes continual learning easier for prior-focused approaches. Continual learning is at its hardest when new tasks resemble old ones, enough that the model makes confident but incorrect predictions on the new data. Prior-focused approaches fail in such cases, and existing evaluations do not capture these scenarios. In this section we will demonstrate these failure cases and others.

3.1 Permuted MNIST

Permuted MNIST represents an unrealistic best case scenario for continual learning. The permutation means that most inputs from the new dataset $\mathcal{D}_{t+1}$ resemble none of the classes from $\mathcal{D}_t$. In practice, this means that during early training on $\mathcal{D}_{t+1}$ the model's strongest predictions have low output probabilities compared with more realistic continual learning setups. As we will see next, Permuted MNIST's setup leads the output layer gradient magnitude to be small, and a consequence the model tends not to forget the previous tasks.

The derivative of each output weight w_k in the output layer of a model is multiplied by a scale given by the cross-entropy loss derivative $\frac{\partial \mathcal{L}}{\partial o_k} = p_k - y_k$ where y_k is the k'th entry of a one-hot vector of labels and p_k is the probability predicted for class k. Assume the ground truth class for a certain input is c. A confident incorrect prediction of class c' would mean large magnitude gradients for the desired unit c which we want to adapt, but also for output unit c' which we do not wish to change. On the other hand, uniform output probabilities reduce the magnitude of the gradient for all units apart from c, spreading the forgetting effect of the training. In the former case, such large magnitude gradients on incorrect classes are the cause of forgetting in neural models. Permuted MNIST's randomly permuted tasks mask this 'forgetting scenario' which is common in many continual learning tasks (e.g. patient records in hospitals are similar to previously observed patients, but might contain different biases between different tasks). The experiment can therefore be seen as 'biased' towards current prior-focused approaches that will otherwise fail when consecutive tasks might be similar enough to give false confidence during training. We will demonstrate these ideas empirically in the next sections.

3.2 Split MNIST

Split MNIST is more challenging than Permuted MNIST insofar as it introduces more realistic differences between tasks. However, the multi-headed version, which is most often used, is not representative of real-world continual learning use-cases. To implement multi-heading, one needs to identify which task each example belongs to. Otherwise one cannot pick a head to train and predict with. Usually, if that were possible, continual learning would be unnecessary, since one could use a separate classifier for each task. Unfortunately, a model which performs well when implemented in a multi-headed way might do very badly when faced with a single-headed scenario. To illustrate, suppose some model has already been trained on the first four tasks of Split MNIST and is then tested on the digit '1'. In the single-headed variant, when shown a '1' it may incorrectly predict the label is seven, which was seen more recently. In the multi-headed variant, we *knowingly assume* that the label comes from [0:1]. Because the model only needs to decide between 0 and 1 (and not even consider if the image is a 7), a multi-headed model could correctly predict the label is 1 even though the same approach will completely fail in a single-headed experiment setup. A multi-headed evaluation can therefore make it seem as if an approach has solved a continual learning problem when it has not. *Single-headed Split MNIST*—an evaluation not currently used in the field—would require a model to work well even when the parameters change significantly to adapt to more than two new and different tasks. Following the same argument as with Permuted MNIST above (and as we will see in the next section), single headed MNIST leads to difficulties in existing approaches because of the destructive gradients in the output layer of the model.

3.3 Other non-representative evaluations

Two-task transfer is not representative of realistic problems because continual learning use-cases often involve a long series of tasks, not just two. An algorithm might have elements that perform well with just one previous task but fail with more.[1] In other cases, experiments revisit each dataset multiple times [14, 29]. This is similarly unreflective of the continual learning setting where old datasets are no longer accessible.

4 Unifying Prior-focused and Likelihood-focused Continual Learning

Before we contrast prior-focused and likelihood-focused approaches in continual learning, let us highlight the tight connection binding the two together. Each of VCL, EWC, and SI effectively set the parameters at the end of the last task as a prior, which is explicit in the Variational Inference (VI) derivation supporting VCL [22]. We approach the derivation of continual learning in VI differently. Instead of changing priors between tasks, we adapt the log-likelihood component of the loss to depend on past datasets. We view estimating the log-likelihood component as using Monte Carlo sampling from densities of past data, which can be modelled using generative models. This offers a principled derivation combining the likelihood-focused approach to continual learning together with the prior-focused ones. In the following we develop the unifying Bayesian framework, focusing on the VI case in order to match the current state-of-the-art algorithm—VCL. An analogous derivation in the maximum likelihood estimation setting would recover an algorithm resembling a more streamlined version of the model DGR proposed by Shin et al. [30].

4.1 A Likelihood-focused Approach using a Variational Free Energy Loss

In non-continual learning, we aim to learn parameters ω using labeled training data $\mathcal{D} = (\mathbf{x}, y)$ to infer $p(y|\omega, \mathbf{x})$. In the continual learning context, the data is not independently and identically distributed (i.i.d.), but instead may be split into separate tasks $\mathcal{D}_t = (\mathrm{X}_t, \mathrm{Y}_t)$ whose individual examples $\mathbf{x}_t^{(n_t)}$ and $y_t^{(n_t)}$ are assumed to be i.i.d. We assume in this case that the tasks are clearly separated, although in general this splitting might be based on unsupervised learning.

Under the standard VI approach [11], we want to find the posterior over the parameters, $p(\omega|\mathcal{D}_{0:T})$ which let us estimate a probability distribution for y. The posterior is generally intractable, so we introduce a tractable approximating distribution $q_\theta(\omega)$ and minimize the Kullback-Leibler divergence

[1] For example, the approximate Fisher information matrix used in EWC is estimated using a Taylor expansion which is only locally accurate in one part of parameter-space. If, after many tasks, the model reaches a very different part of parameter-space, the estimated Fisher information for old tasks will be inaccurate.

between the approximating distribution and posterior:

$$q_\theta^* = \underset{q_\theta \in \mathcal{Q}}{\operatorname{argmin}} KL\big(q_\theta(\omega) \parallel p(\omega|\mathcal{D}_{0:T})\big) \tag{1}$$

which, after applying Bayes' theorem and removing constant terms, becomes:

$$q_\theta^* = \underset{q_\theta \in \mathcal{Q}}{\operatorname{argmin}} \Big(KL\big(q_\theta(\omega) \parallel p(\omega)\big) - \mathbb{E}_{\omega \sim q_\theta(\omega)} \Big[\log p(\mathrm{Y}_{0:T}|\omega, \mathrm{X}_{0:T}) \Big] \Big). \tag{2}$$

This is equivalent to minimizing the variational free energy of the model. The first term can be interpreted as pushing q towards a prior about the distributions of the parameters. The second term is data-dependent and is the expected negative log-likelihood of the observed data given the parameters of the model. To extend this log-likelihood term to the continual learning setting, we observe that:

$$\log p(\mathrm{Y}_{0:T}|\omega, \mathrm{X}_{0:T}) = \sum_{t=1}^{T} \log p(\mathrm{Y}_t|\omega, \mathrm{X}_t) = \sum_{t=1}^{T} \sum_{n=1}^{N_t} \log p(y_t^{(n)}|\omega, \mathbf{x}_t^{(n)}). \tag{3}$$

Using 2 and 3 and rearranging gives an expression for the variational free energy $\mathcal{F}_T$ that can be used to train on dataset $\mathcal{D}_T$ and can be decomposed into a sum of separate terms for each dataset $\mathcal{D}_{0:T}$:

$$\mathcal{F}_T \propto \Big(KL\big(q_{\theta_T}(\omega) \parallel p(\omega)\big) - \sum_{t=1}^{T} \mathcal{L}_t \Big) \propto \sum_{t=1}^{T} \Big(\frac{1}{T} KL\big(q_{\theta_T}(\omega) \parallel p(\omega)\big) - \mathcal{L}_t \Big) \tag{4}$$

where $\mathcal{L}_t$ is the expected log-likelihood over $\mathcal{D}_t$:

$$\mathcal{L}_t = \sum_{n=1}^{N_t} \mathbb{E}_{\omega \sim q_{\theta_T}(\omega)} \left[\log p(y_t^{(n)}|\omega, \mathbf{x}_t^{(n)} \right]. \tag{5}$$

We cannot estimate $\mathcal{L}_t$ for $t < T$ directly because we have discarded those datasets. Instead, we approximate the sum in (5) as an integral over a data distribution $p_t(\mathbf{x}, y)$ which is in turn approximated by a generative model. That is, we can approximate equation (5) as:

$$\mathcal{L}_t \approx \frac{1}{N_t} \int \log \big[p(y|\omega, \mathbf{x}) \big] p_t(\mathbf{x}, y) q(\omega) d\mathbf{x} dy d\omega. \tag{6}$$

In order to approximate the distribution of past datasets $p_t(\mathbf{x}, y)$, we train a GAN $q_t(\mathbf{x}, y)$ [7] to produce $(\hat{\mathbf{x}}, \hat{y})$ pairs for each class in each dataset as it arrives. After that dataset is used, the data are discarded and the generator is kept.

Data from the distributions $\hat{\mathbf{x}}, \hat{y} \sim q_{1:T-1}(\mathbf{x}, y)$ can supplement the actual data for task T to create $(\tilde{\mathbf{x}}, \tilde{y}) = (\hat{\mathbf{x}} \cup \mathbf{x}, \hat{y} \cup y)$. Because we were able to separate (4) as an sum of task-specific terms we can minimize it by minimizing:

$$\tilde{\mathcal{F}} = \mathbb{E}_{\omega \sim q_{\theta_T}(\omega)} \big[\log p(\tilde{y}|\omega, \tilde{\mathbf{x}}) \big] - \frac{1}{NT} KL\big(q_{\theta_T}(\omega) \parallel p(\omega)\big) \tag{7}$$

which we approximate by averaging over sampled mini-batches. We refer to this model as VGER, given in Algorithm 1. This model can be seen as an exemplar of the family of likelihood-focused approaches. By setting the prior to be given as the posterior from the previous task, or objective can further be seen as merging such likelihood-focused approaches together with prior-focused approaches.

4.2 Comparison Between VGER and Other Likelihood-focused Approaches

Existing works already use generative models to simulate past datasets as a tool for continual learning [27, 30]. Our derivation provides a unified view for these algorithms together with prior-focused approaches. Additionally, our suggested VGER is novel in its application of variational inference and, because it was derived from first principles, simpler in design compared to [30] for example. VGER is designed to be as analogous as possible to VCL—the current state-of-the-art—which makes it ideally suited for experimental comparison. Further, VGER's probabilistic nature allows us to identify new tasks by examining model uncertainty (see §6.3).

Many alternative ways of estimating density models for use in generating data are possible. One could use conditional GANs using the training class label as an input. Or one might follow Shin

Algorithm 1 VGER

Input: Prior $p(\omega)$, inputs to generate per class n.
Output: Variational and predictive distributions $q_{\theta_t}(\omega), p(\hat{y}|\omega_t, \hat{\mathbf{x}})$ for $0 \leq t < T$.
Initialize $\omega \leftarrow \mathcal{N}(0, 1)$.
for $t = 0$ **to** T **do**
  Observe dataset $\mathcal{D}_t$ and initialize $\mathcal{G}_t \leftarrow \emptyset$ and $q_{\theta_0}(\omega) \leftarrow p(\omega)$.
  for $t' = 0$ **to** t **do**
    Sample n replay from $\text{Gen}_{t'}$ as $\mathcal{R}_{t'}$
    $\mathcal{G}_{t'} \leftarrow \mathcal{G}_{t'-1} \cup \mathcal{R}_{t'}$
  end for
  $\omega \leftarrow \text{argmin}(\tilde{\mathcal{F}})$ from equation (7) on $\mathcal{D}_t \cup \mathcal{G}_t$.
  Train and store Gen_t on $\mathcal{D}_t$.
end for

	Entropy
Permuted	0.003
Split	0.453

Figure 1: Average entropy of predictions on Task B, early in training, for both Permuted MNIST and Split MNIST; Note the 2 orders of magnitude difference between the two settings.

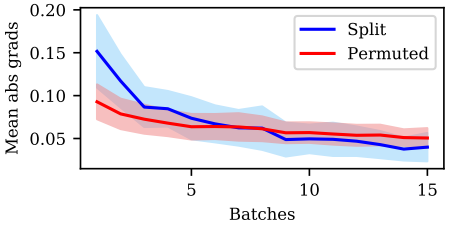

Figure 2: Early in training Task B, gradients are unusually low in the Permuted setting because permuted digits are unrecognizable. This makes continual learning unrealistically easy in this evaluation. Averaged over 100 runs, shading reflects standard deviation.

Model	AUC
VCL	0.76
VGER	1

Figure 3: Area under the curve (AUC) for ROC plot of correct detections of new datasets using model uncertainty. Higher is better calibrated. Each ROC plot (given in appendix B.6) compares various thresholds θ of declaring 'the uncertainty has crossed the threshold; this is a new dataset!'.

et al. [30] whose 'scholar' uses old classifiers to label unconditional GAN outputs.[2] Coresets or randomly preserving a subset of examples are other ways to estimate the density model. Whether coresets or generative models are more computationally efficient, or even tractable, depends on the data-space. However, we find that VGER is not very sensitive to the length of training of the generative model and even poor generative models preserve classifications adequately (see figure 7). Generative models can find complex data-spaces difficult, though major advances have been made with complex environments [32]. Moreover, generative models allow for differential privacy guarantees that coresets do not [23].

5 Empirical Analysis of Existing Evaluations

We next offer experimental support for our claims above. The aim of this section is twofold. First, we demonstrate failure cases of existing experimental evaluations. Second, we demonstrate that likelihood-focused approaches perform as well as prior-focused approaches on current evaluations. For this we use VCL (the current SOTA prior-focused approach), and our suggested VGER (the closest appropriate likelihood-focused counterpart to VCL, for the sake of a fair comparison with VCL). Later, in §6, we will introduce new evaluations which highlight failure cases of existing prior-focused *models*.

5.1 Shortcomings of Permuted MNIST

We introduce two tests to show the shortcomings of the Permuted MNIST experiment discussed in §3.1. For each test we consider both the Permuted and Split settings. In each case, we train a model on Task A before training on Task B. In the split setting, Task A is the first five digits of MNIST

[2]DGR highlights one of the benefits of the simpler, non-recursive, generative model we use. For DGR, performance degrades gradually because labelling errors accumulate. If the first classifier has 99% accuracy, then 1% of the labels for generated examples used to train the second task are wrong. Over a long sequence of tasks, these errors build up.

and Task B the last five digits. In the permuted setting, Task A is MNIST and Task B is a random permutation of the pixels of MNIST.

We first validated our hypothesis that predictions are much more uniform in the Permuted setting. For this we evaluate the entropy of the output probability vector. Low entropy indicates a confident prediction of a single class, whereas a high entropy indicates a 'uniform' prediction, spreading the mass across different classes. These are summarised in Table 1, supporting our claim. Next, we test our hypothesis from §3.1 that more 'uniform' predictions lead to slower 'catastrophic forgetting'. To test this hypothesis, after training on Task A, we measured gradients for the final weight layer while training Task B in the two variants (figure 2). All results are averaged over 100 runs. Full experimental details are in Appendix B.1. We found that having digits similar to previously observed ones in the *Split setting* led to gradients having much higher magnitude compared to images not resembling previously observed ones in the *permuted setting* for early batches. Over time, the gradients levelled out for both. This shows that the Permuted MNIST task is unrepresentative of the continual learning setting, because the permuted digits look so unrecognizable that they create less disruptive gradients, leading to artificially lessened forgetting.

5.2 Evaluating VGER and VCL with Permuted MNIST

We find that on Permuted MNIST, described in §3.1, VGER performs marginally better than VCL with a coreset (figure 4). This configuration of VCL was previously shown to outperform EWC and SI [22]. This demonstrates that observation-space approaches to continual learning perform competitively even on existing evaluations. Details of training hyper-parameters can be found in Appendix B.2 and were chosen to match all stated parameters in Nguyen et al. [22].

5.3 Evaluating VGER and VCL with Multi-headed Split MNIST

As we showed in §5.1, the Permuted MNIST task is unrealistic because the tasks are too independent. Split MNIST, described in §3.2, has tasks that are similar but different enough to confuse neural networks. We begin by evaluating models with multi-heading. That is, each prediction is the most probable class out of the classes of the task being evaluated. In addition, following Zenke et al. [34] and Nguyen et al. [22], VCL is trained with five separate heads which make predictions over only two classes, which are trained separately but share hidden layers.

We find that VGER performs similarly to VCL in multi-headed Split MNIST (figure 5). VGER starts to edge out VCL by the last task, consistent with slightly better multi-task performance. Intriguingly, a model which does not use the VCL loss but uses only coresets performs similarly to the VCL loss combined with coresets. Coresets, as used by Nguyen et al. [22], can be thought of as a very restricted likelihood-based approach. Details of training hyperparameters can be found in Appendix B.4 and were chosen to match all stated parameters in Nguyen et al. [22].

6 Proposed Evaluations for Future Continual Learning Research

The aim of this section is to demonstrate failure cases and limitations of prior-focused approaches. Specifically, we highlight and contrasts these to likelihood-focused approaches which do not share the same failure modes. Both VGER and VCL perform well on the experimental evaluations from §5. In this section, we evaluate both VCL and VGER on a modification of Split MNIST, by simply removing multi-heading, and show that VGER outperforms VCL by a wide margin. As we argued in §3.2, single-headed models are more reflective of the use-cases that motivate continual learning. We then introduce a new version of single-headed Split MNIST which reflects the importance of running time in cases where continual learning is necessary because retraining on all old data is impractical due to time constraints (e.g. in real-time systems where the system dynamics changes faster than the time it would take to retrain). We name this setup *Timed Split MNIST*. Lastly, we assess the quality of our models' uncertainty estimates, which can be used to decide when to start a new 'retraining phase' (for models that want to reason about whether they are facing an unfamiliar task, among other safety concerns). We believe these new evaluations to be as important as continuing to scale up to larger, more complex, environments.

6.1 Single-headed Split MNIST

We use the single-headed variant of Split MNIST to represent the case where it is not possible to select from multiple models at evaluation time. We perform exactly the same experiment as in the

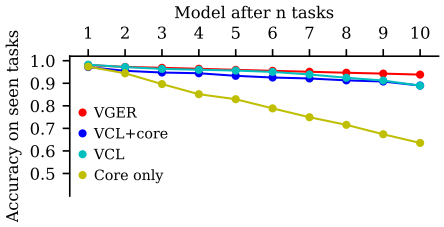

Figure 4: **Permuted MNIST.** VGER has marginally better performance than VCL with or without a coreset, which has been shown to outperform EWC and SI. Coresets without the VCL loss performs worse. Averaged over 10 runs.

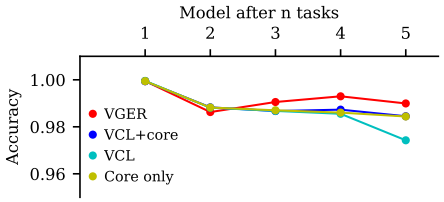

Figure 5: **Multi-headed Split MNIST.** VGER performs similarly to VCL, which has been shown to outperform EWC and SI. VCL appears to work largely independently of coresets, and coresets work well on their own. Averaged over 10 runs.

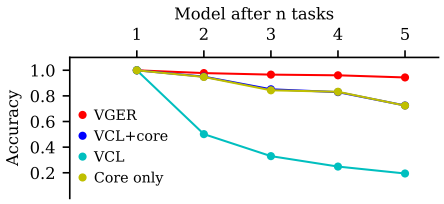

Figure 6: **Single-headed Split MNIST.** VGER outperforms VCL with coresets. Coresets seem to contribute much of VCL's performance, as coresets alone performs identically. VCL alone forgets old tasks completely. Likelihood-focused approaches seem better suited to the single-headed case. Averaged over 10 runs.

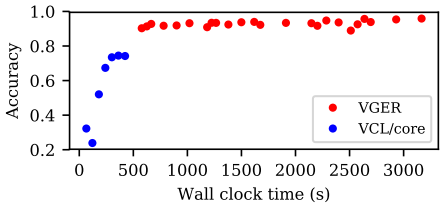

Figure 7: **Timed Single-headed Split MNIST.** Hyperparameter choice can lead to slower, more accurate, models for each architecture. Ideal performance is in the top left corner. VGER is accurate but slower than VCL.

previous section, except that predictions are not constrained to be from the 'ground truth' task. If a network predicts with high output probability that any input is most likely to be from the most recent task, it will now have 0% accuracy on old tasks, even if it could still distinguish zeros and ones (having been told to choose one of the two). For VCL, we remove the multiple heads used for training, so that all training is on a single head with ten outputs. Nguyen et al. [22] have already shown that EWC performs poorly in single-headed Split MNIST, though they did not evaluate VCL in this way. In this new setting, VGER outperforms VCL by a wide margin (figure 6).

6.1.1 Examining Coresets in VCL

In our experiments, we further examined a variant of VCL that uses coresets in exactly the same way as Nguyen et al. [22] but uses an ordinary variational free energy loss rather than the VCL loss (i.e. without refreshing the prior after each task). This allows us to estimate how much of the performance of VCL with coresets comes from VCL and how much comes from the coresets.[3] We find that the model using coresets alone performs much better than VCL alone in single-headed Split MNIST (figure 6). This suggest that coresets, related to our likelihood-focused approach, play a big role in the combined performance of VCL and coresets, especially in a single-headed setting. In fact, VCL with coresets can be seen as a hybrid prior-likelihood-focused approach.

6.2 Timed Single-headed Split MNIST

We use *Timed Split MNIST* to evaluate the trade-off of training time against accuracy in continual learning. This experiment is important because much of the need for continual learning comes from situations where a real-time system must respond to distributional shift quickly enough that it is impractical to retrain on all previous data. Continual learning systems generally have hyperparameters that affect both performance and speed. For example, in VGER, training on larger sampled datasets improves performance but takes more time. In figure 7 we show the average accuracy of various models plotted against the wall-clock time taken to finish training for a wide range of such hyper-

[3] Nguyen et al. [22] compares VCL to a hypothetical model that only uses a coreset. However, in their work, the 'coresets-only' model performs much worse than VCL even on the first task, where no continual learning is happening. This suggests the model is seeing *only* coresets—much less data. Instead, we compare VCL with coresets and fine-tuning to a model that is identical but uses a VFE loss instead of a VCL loss.

parameters. The objective is not to carry out an exhaustive hyper-parameter search, but to indicate different possibilities for performance/time trade-offs. Details can be found in Appendix B.5 along with a multi-headed version. None of the configurations of VCL provides good accuracy on the single-headed task. At the same time, none of the configurations of VGER is able to provide good time performance (see appendix B.5.1). We highlight these shortcomings for future research to address.

6.3 Model Uncertainty on Unseen Tasks

Model uncertainty gives another tool for understanding forgetting. A model with well-calibrated uncertainty enables detection of new distributions or the escalation of anomalous examples to human oversight. A good continual learning model should tend to be more certain on tasks it has seen before than tasks it has not seen. We train VGER and VCL on the single-headed Split MNIST task as before. After training on each task, we measure the model's uncertainty when shown data from each task, both seen and unseen. We then use the uncertainty information to predict whether the model has seen the dataset before or not. If the uncertainty is higher than a certain threshold, we announce 'the model was not trained on this dataset!'. We evaluated this with different thresholds, generating an ROC plot (full experiment setup with plots is given in appendix B.6). We compare the AUC of both these ROC plots in Table 3. As can be seen, VGER is able to correctly detect all tasks it has not seen before, whereas VCL fails on a considerable number of those. A more thorough experimental comparison (with additional experiments) is given in the appendix due to space constraints.

7 Discussion

Simple experimental evaluations shape the field of continual learning by providing reproducible comparisons between architectures and testing whether an architecture satisfies minimal objectives. Blindspots in existing evaluations have led to inadequate comparison in recent research. We showed that existing evaluations are biased towards prior-focused approaches, which perform badly in modified evaluations which require parameters to change significantly between tasks. We advocate for the likelihood-focused approach, and provided a theoretical foundation for likelihood-focused continual learning combined with prior-focused ones. We implemented the likelihood-focused approach as a VI model VGER which outperforms the state-of-the-art prior-focused model on more representative experiments—single-headed Split MNIST, Timed Split MNIST, and model uncertainty on unseen tasks relative to seen ones. But many challenges remain. No existing approach has been shown to succeed in very complex settings yet. We hope that by selecting the right targets for experimental progress and drawing on both prior- and likelihood-focused methods the community will be able to create models that ultimately extend to more complex, real-world, benchmarks.

Acknowledgements

We would like to thank for their advice and comments on drafts of this paper: Owain Evans, Gregory Farquhar, Yingzhen Li, Jelena Luketina, Cuong Viet Nguyen, and Richard Turner. This research was supported by The Alan Turing Institute and the EPSRC grant number EP/P00881X/1 at the Centre for Doctoral Training in Cyber Security at the University of Oxford. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

[1] Blundell, Charles, Cornebise, Julien, Kavukcuoglu, Koray, and Wierstra, Daan. Weight Uncertainty in Neural Networks. *Proceedings of the 32nd International Conference on Machine Learning*, 37:1613–1622, 2015.

[2] Donahue, Jeff, Jia, Yangqing, Vinyals, Oriol, Hoffman, Judy, Zhang, Ning, Tzeng, Eric, and Darrell, Trevor. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. *Proceedings of the 31st International Conference on Machine Learning*, 31: 647–655, 2014.

[3] Fernando, Chrisantha, Banarse, Dylan, Blundell, Charles, Zwols, Yori, Ha, David, Rusu, Andrei A., Pritzel, Alexander, and Wierstra, Daan. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. *arXiv*, 2017.

[4] Gal, Yarin. Uncertainty in Deep Learning. *PhD Thesis*, 2016.

[5] Gal, Yarin and Ghahramani, Zoubin. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *Proceedings of the 33rd International Conference on Machine Learning*, 48:1050–1059, 2015.

[6] Gal, Yarin and Ghahramani, Zoubin. Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference. *4th International Conference on Learning Representations workshop track*, 2016.

[7] Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative Adversarial Nets. *Advances in Neural Information Processing Systems 27*, pp. 2672–2680, 2014.

[8] Goodfellow, Ian J., Mirza, Mehdi, Xiao, Da, Courville, Aaron, and Bengio, Yoshua. An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. *arXiv*, 2013.

[9] Graves, Alex. Practical Variational Inference for Neural Networks. *Neural Information Processing Systems*, 2011.

[10] Huszár, Ferenc. Note on the quadratic penalties in elastic weight consolidation. *PNAS*, 115(11): E2496–E2497, 2018.

[11] Jordan, Michael I., Ghahramani, Zoubin, Jaakkola, Tommi S., and Saul, Lawrence K. Introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[12] Jung, Heechul, Ju, Jeongwoo, Jung, Minju, and Kim, Junmo. Less-forgetting Learning in Deep Neural Networks. *arXiv*, 2016.

[13] Kingma, Diederik P., Salimans, Tim, and Welling, Max. Variational Dropout and the Local Reparameterization Trick. *Advances In Neural Information Processing Systems*, pp. 2575–2583, 2015.

[14] Kirkpatrick, James, Pascanu, Razvan, Rabinowitz, Neil, Veness, Joel, Desjardins, Guillaume, Rusu, Andrei A., Milan, Kieran, Quan, John, Ramalho, Tiago, Grabska-Barwinska, Agnieszka, Hassabis, Demis, Clopath, Claudia, Kumaran, Dharshan, and Hadsell, Raia. Overcoming catastrophic forgetting in neural networks. *Neural Information Processing Systems*, 114(13): 3521–3526, 2017.

[15] LeCun, Yann, Bottou, Leon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *IEEE*, 86(11):2278–2324, 1998.

[16] Lee, Sang-woo, Kim, Jin-hwa, Jun, Jaehyun, Ha, Jung-woo, and Zhang, Byoung-tak. Overcoming Catastrophic Forgetting by Incremental Moment Matching. *Advances in Neural Information Processing Systems*, 2017.

[17] Li, Zhizhong and Hoiem, Derek. Learning without Forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[18] Lopez-Paz, David and Ranzato, Marc'Aurelio. Gradient Episodic Memory for Continual Learning. *Proceedings of the 31st Conference on Neural Information Processing Systems*, 31, 2017.

[19] MacKay, David J. C. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3):448–472, 1992.

[20] Neal, Radford M. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995.

[21] Netzer, Yuval, Wang, Tao, Coates, Adam, Bissacco, Alessandro, Wu, Bo, and Ng, Andrew Y. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[22] Nguyen, Cuong V., Li, Yingzhen, Bui, Thang D., and Turner, Richard E. Variational Continual Learning. *International Conference on Learning Representations*, 2018.

[23] Papernot, Nicolas, Abadi, Martín, Erlingsson, Úlfar, Goodfellow, Ian, and Talwar, Kunal. Semi-supervised knowledge transfer for deep learning from private training data. *Proceedings of the 5th International Conference on Learning Representations*, 2017.

[24] Parisi, German I., Kemker, Ronald, Part, Jose L., Kanan, Christopher, and Wermter, Stefan. Continual Lifelong Learning with Neural Networks: A Review. *arXiv*, 2018.

[25] Razavian, Ali Sharif, Azizpour, Hossein, Sullivan, Josephine, and Carlsson, Stefan. CNN features off-the-shelf: An astounding baseline for recognition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 512–519, 2014.

[26] Ritter, Hippolyt, Botev, Aleksandar, and Barber, David. Online Structured Laplace Approximations For Overcoming Catastrophic Forgetting. *arXiv*, 2018.

[27] Robins, Anthony. Catastrophic Forgetting, Rehearsal and Pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

[28] Rusu, Andrei A., Rabinowitz, Neil C., Desjardins, Guillaume, Soyer, Hubert, Kirkpatrick, James, Kavukcuoglu, Koray, Pascanu, Razvan, and Hadsell, Raia. Progressive Neural Networks. *arXiv*, 2016.

[29] Schwarz, Jonathan, Luketina, Jelena, Czarnecki, Wojciech M., Grabska-Barwinska, Agnieszka, Teh, Yee Whye, Pascanu, Razvan, and Hadsell, Raia. Progress & Compress: A scalable framework for continual learning. *arXiv*, 2018.

[30] Shin, Hanul, Lee, Jung Kwon, Kim, Jaehong, and Kim, Jiwon. Continual Learning with Deep Generative Replay. *Advances In Neural Information Processing Systems*, pp. 2994–3003, 2017.

[31] Srivastava, Rupesh Kumar, Masci, Jonathan, Kazerounian, Sohrob, Gomez, Faustino, and Schmidhuber, Jürgen. Compete to Compute. *Advances in Neural Information Processing Systems*, pp. 2310–2318, 2013.

[32] Wang, Ting-Chun, Liu, Ming-Yu, Zhu, Jun-Yan, Tao, Andrew, Kautz, Jan, Catanzaro, Bryan, Zhu, Ming-yu Liu Jun-yan, Tao, Andrew, Kautz, Jan, and Nov, C V. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. *arXiv*, 2017.

[33] Yosinski, Jason, Clune, Jeff, Bengio, Yoshua, and Lipson, Hod. How transferable are features in deep neural networks? *Advances In Neural Information Processing Systems*, pp. 3320–3328, 2014.

[34] Zenke, Friedemann, Poole, Ben, and Ganguli, Surya. Continual Learning Through Synaptic Intelligence. *Proceedings of the 34th International Conference on Machine Learning*, pp. 3987–3995, 2017.

Appendices

A Further Prior Work

A.1 Elastic Weight Consolidation

At the end of training on each dataset, EWC [14] estimates the contribution each parameter makes to the gradient of the loss by approximating the Hessian of the likelihoods with:

$$F_t \propto \sum_{n=1}^{N_t} \left(\nabla_\omega \log p(y_t^{(n)} | \omega, \mathbf{x}_t^{(n)}) \right)^2 \Big|_{\omega=\omega_t}$$

This approximate Fisher information matrix is estimated at the end of each task. The authors state that in the multi-task case the regularization term for task T is the sum of separate regularization terms for each past dataset $\mathcal{D}_t$,[4]

Kirkpatrick et al. [14] show EWC works well on the Permuted MNIST task introduced by Goodfellow et al. [8]. They also show that EWC reduces forgetting on a succession of Atari games.

A.2 Synaptic Intelligence

Published soon after EWC, SI has a similar loss but computes the contribution of each parameter to the gradient of the loss over the entire course of training. The authors explicitly acknowledge that the derivation of their regularization term only extends to the two-task case, but point out that the model performs well even in multi-task settings.

Zenke et al. [34] show that SI works in a *multi-headed* version of the Split MNIST task which they introduce, and matches EWC on the Permuted MNIST task. They also show some transfer learning on a multi-headed split CIFAR task of their own design.

A.3 Variational Continual Learning

Nguyen et al. [22] motivate their regularization through VI. They argue that the posterior of the parameter distribution after learning a task should be the prior when learning the next task. They calculate the Kullback-Leibler (KL) divergence between the current distribution and the previous posterior. This is in contrast with the more usual Variational Free Energy (VFE) loss which uses the KL divergence to a constant prior that is not updated after each task. The KL divergence can be represented approximately as a quadratic regularization with rotation, just like the two approaches above, as Nguyen et al. [22] point out.

In addition to the Variational Continual Learning (VCL) regularization term, Nguyen et al. [22] add a coreset which samples examples from old datasets, which are withheld from the main dataset. Rather than discarding all of an old dataset, as EWC and SI do, only the vast majority (about 99.7%) is discarded. Before being evaluated, the model is trained against the coreset. For multi-headed settings (see §3.2) coresets train each head separately.

Nguyen et al. [22] show that VCL with or without the use of coresets outperforms both SI and EWC on the Permuted MNIST task. They show that VCL outperforms EWC and SI on the multi-headed Split MNIST task and on a very similar task with notMNIST. They take advantage of the VI setting to show reduced uncertainty in a generative task based on MNIST and notMNIST.

A.4 Other Observation-space Approaches

Some observation-space approaches have been proposed before. For example,Shin et al. [30] introduce Deep Generative Replay (DGR). They use generative models to supplement training, inspired by neuroscientific analogy. Shin et al. [30] show DGR's performance on Permuted MNIST as well as on Split MNIST, similarly to the three prior-focused papers above. In addition, they demonstrate their performance on a two-task evaluation, switching from classifying MNIST to SVHN [21] and back.

[4] Huszár [10] suggests that one should only regularize relative to the most recent weights, because it incorporates the old information, but using all historic Fisher information matrices.

DGR is closely related to our suggested VGER. We do not intend VGER as a replacement for DGR, instead we see VGER as an exemplar of the likelihood-focused approach. Nevertheless, VGER is more principled and streamlined.

A.5 Dynamic architecture approaches

Other work, which is not our focus, has also made progress with continual learning. Progressive neural networks [28] conditions the newest model on the outputs of old, stored, models. Some approaches split the network into regions that have different roles. In Li & Hoiem [17] one set of shared parameters governs a feature extraction component while each task is given parameters on top of that component. During training, the combined shared and old task networks are encouraged to classify similarly to a stored old version of the model, while the combined shared and new task networks are trained on the new task. Jung et al. [12] stochastically freeze the final layer to encourage lower layers to extract features from all tasks that the final layer can use to classify. Others have semi- or fully-fixed a shared part of a network with task-specific layers on top [2, 25, 33]. PathNet uses evolutionary selection to try to learn which patches of a larger network are helpful to each task rather than making a layer-by-layer assumption [3]. This broad family of approaches are especially useful in two-task transfer cases, but it can become impractical to introduce the many task-specific weights in the more general continual learning case that we consider.

Others find that adjustments to learning dynamics can reduce catastrophic forgetting, presumably by encouraging networks to use their capacity fully. This family of approaches includes selecting activations and using dropout to reduce forgetting [8, 31]. These are useful, but do not go far enough to solve forgetting fully. Lee et al. [16] extend dropout by shifting the zero-point of each dropout parameter to the parameter value from training on the previous task.

B Experimental Details and Further Figures

B.1 Shortcomings of Permuted MNIST Experimental Settings

To measure the gradients during training, we first trained on Task A for 120 epochs with batch size 256 and an Adam optimizer. We then trained on Task B for 15 batches of 16 digits each. We measured the average absolute value of the gradients. We averaged over 100 training runs for each setting (with a different permutation each time in the Permuted setting), resetting the model to its initial position after each run. Graphs show the standard deviation of the average gradient of each batch. Here we trained an MLP with two hidden layers of 256 units. We used an Adam optimizer with learning rate 10^{-3} and batch size 256 for 30 epochs.

B.2 Permuted MNIST Details

For permuted MNIST, we follow Nguyen et al. [22] where possible. We use a Bayesian neural network [9] with two hidden layers of 100 units with ReLu activations. The priors are initialized as a unit Gaussian and the parameters are initialized with the mean of a pre-trained maximum likelihood model and a small initial variance (10^{-6}). We use the Adam optimizer [13] with learning rate 10^{-3}. For both VGER and VCL we use a single head, again following Nguyen et al. [22]. For all results we present the average over 10 runs, with a different permutation each time. Standard errors are not shown as they are under a tenth of a percent.

For VCL we train for 100 epochs using a batch size of 256 on all the data except the with-held coresets. We train the whole single head on coresets for 100 epochs and use 200 digits of each permutation as a coreset chosen using the same k-center coresets algorithm used by Nguyen et al. [22]. VCL without coresets is exactly the same, but without a final training step on coresets. The coresets only algorithm is exactly the same as VCL, except that the prior is always initialized as though it were the first task.

We train VGER for 120 epochs using a GAN trained for 200 epochs on each MNIST digit. We use 6000 generated digits per class, sampled fresh for each task, and initialize network weights using the previous task. We use a batch size of 256 times the number of seen tasks, ensuring that the number of batches is held constant.

The GAN is trained with an Adam optimizer with learning rate $2 * 10^{-4}$ and β_1 of 0.5. The network has four fully-connected hidden layers with 256, 512, 1024 and 784 weights respectively. It uses Leaky ReLu with α of 0.2.

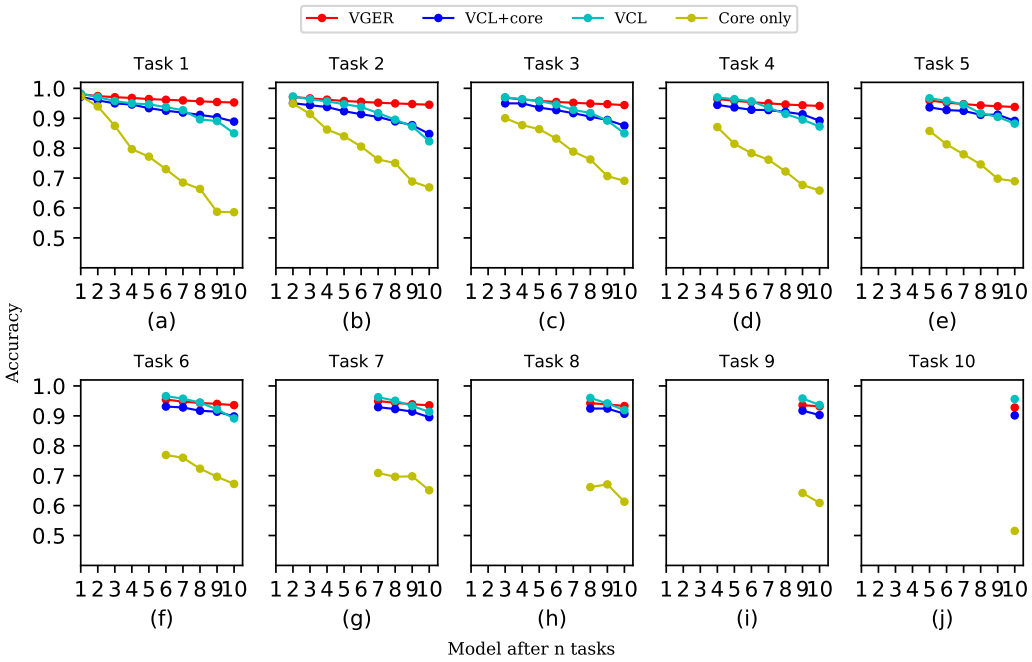

Figure 8: **Permuted MNIST.** VGER performs slightly better than VCL on Permuted MNIST, which has been shown to outperform EWC and SI. Each sub-plot shows performance on one of the tasks for each model that was evaluated on it. After training on each dataset, the model is evaluated on each of the tasks it has seen before. All models are evaluated against Task 1, but only the tenth model is evaluated on all ten tasks.

Below, we further show the performance of each iteration of the continual learning model on each task (figure 8). Each subplot shows just one task. It shows the performance of all models that have seen that task. All models see, and so are evaluated on, Task 1. But only the model that has trained on all tasks is evaluated on Task 10. Models forget gradually and cumulatively, which is somewhat consistent with the reduced disruption that the gradients in Permuted MNIST might be expected to produce.

B.3 Single-headed Split MNIST

The settings for Split MNIST follow Nguyen et al. [22] where possible. We use exactly the same architecture as for Permuted MNIST, including the single head, except that each hidden layer has 256 weights, similarly to Nguyen et al. [22] on their multi-headed Split MNIST. Results are shown averaged over 10 runs, with a different coreset selection each time. Standard errors are not shown as they are of the order of a tenth of a percent.

For all architectures we train for 120 epochs. For VCL we use batch sizes equal to the training set size. We use coresets of digits per task selected using the same k-center coreset algorithm as Nguyen et al. [22], which are withheld from the training. We train for 120 epochs on the concatenated coreset across all the heads together.

For VGER, we use 6000 digits per class generated by a convolutional GAN. Unlike VCL, we cap batch sizes at 30,000 rather than having the batch size equal the training set size. This is because the examples generated by VGER result in larger training sets which exceeded the memory available on our GPU.The GAN is trained for 50 epochs on each MNIST class using the same optimizer as the non-convolutional GAN used for Permuted MNIST. It has a fully connected layer followed by two convolutional layers with 64 and 1 channel(s) and 5x5 convolutions. Each convolutional layer is preceded by a 2x2 up-sampling layer. The activations are Leaky ReLu's with α of 0.2.

In figure 9 we can see the performance of each model on every task. Each subplot shows one of the five Split MNIST tasks. It shows the accuracy of each model tested on that task. We see that for models other than VGER, forgetting happens immediately on each new task and does not recover, and that coresets are essential to good performance.

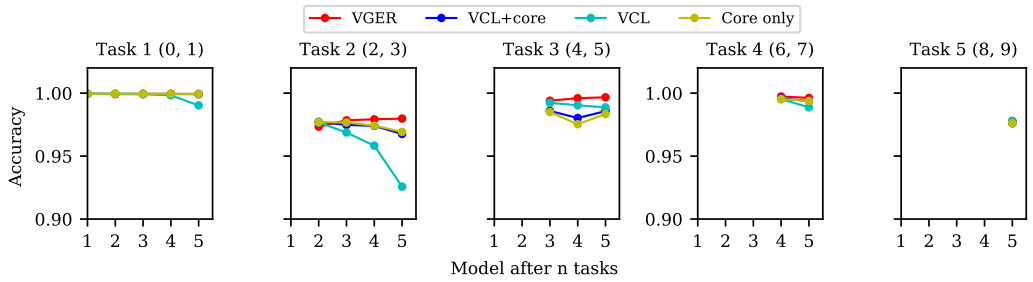

Figure 9: **Multi-headed Split MNIST.** VGER slightly outperforms VCL on multi-headed Split MNIST. We show the accuracy of models after each dataset is shown for each task in turn. We can also see that even without using VCL, a coreset of 40 digits from each class almost matches VCL's performance including a coreset. In the multi-headed setting, predictions are constrained to be of the correct task.

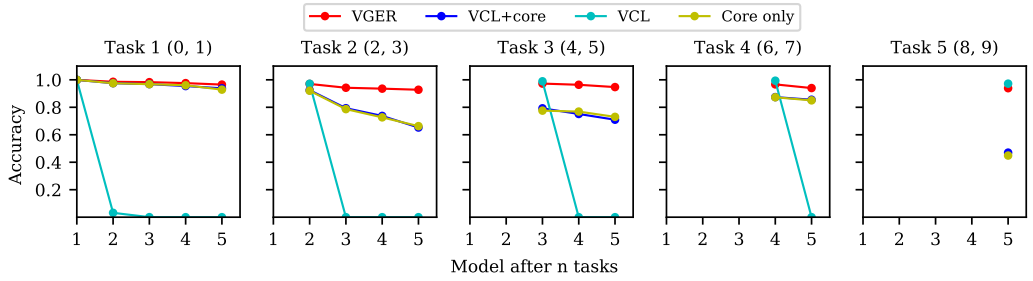

Figure 10: On individual tasks that for VCL performance drops immediately after the new dataset arrives and never recovers.

B.4 Multi-headed Split MNIST

Multi-headed Split MNIST has almost precisely the same settings as the Single-headed Split MNIST above. Following Nguyen et al. [22], for VCL we have five heads in the final layer, each of which is trained separately. Coresets are used to train each head in turn. Batch size equal is to training set size. Results are shown averaged over 10 runs with fresh coreset selection each time. Standard errors are not shown as they are of the order of a tenth of a percent.

For VGER, unlike VCL, we only perform multi-heading as a way of selecting predictions, rather than using multiple heads during training as well, because each batch contains data from multiple tasks.

B.4.1 Using Monte Carlo Dropout

In addition to Bayesian neural networks, we consider the use of Monte Carlo Dropout [5]. We found that this model produced exactly the same performance as a Bayesian neural network, shown in figure 11. However, it could scale effectively to larger images where the Bayesian neural network would struggle.In contrast, we could not perform VCL with MC Dropout because it was not possible to make the parameter-uncertainties specific enough.

For a simple demonstration, we train a network with two convolutional layers with dropout (3x3 kernels and 32 then 64 channels) [6], a 2x2 max pooling layer, and two fully connected layers with dropout. Dropout was 0.25 and an L2 regularization with weight 0.01 produced the KL divergence.

B.4.2 Sensitivity Analysis for VGER Hyper-parameters

We tested VGER with a wide range of hyperparameters and found that it was not very sensitive to them. Training anywhere between 20 and the 120 epochs used by VCL seemed to produce very similar performance (figure 12. Similarly, using anywhere between 2000 and 6000 digits per class of MNIST (MNIST itself has roughly 6000 digits per class) gave similar performance. Below 1000, performance fell (figure 13).

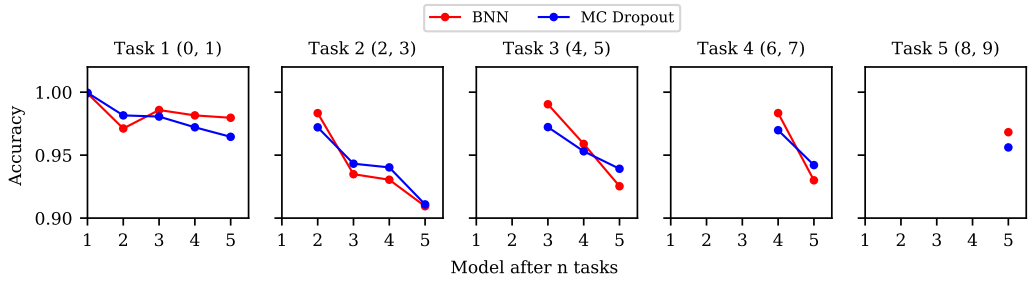

Figure 11: We can achieve the same performance using a Monte Carlo Dropout implementation of VI as we get with a Bayesian neural network. Here we use a simple CNN with two convolutional layers, one pooling layer, and two dense layers, with dropout throughout trained for 25 epochs.

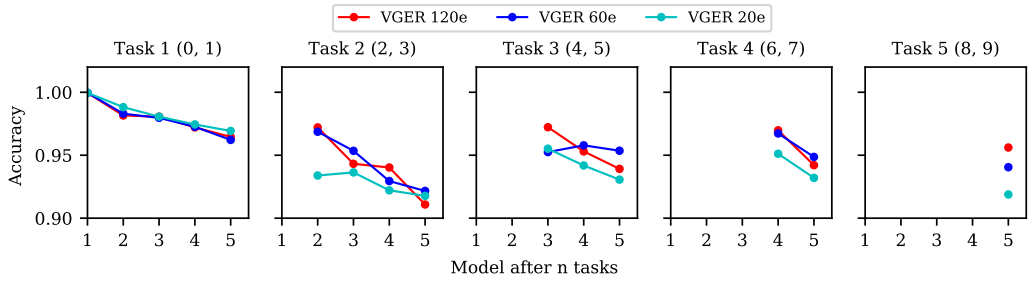

Figure 12: Task accuracy is not strongly affected by further training on single-headed Split MNIST after 20 epochs.

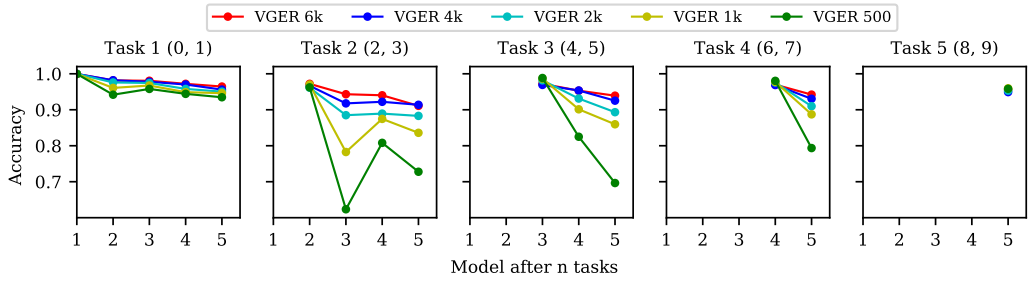

Figure 13: We find that task accuracy is not strongly affected by the number of generated images used, and that using just 2000 is enough to produce good continual learning. But the model performs best when given data matching the original number of digits in the task. For smaller generated sets we train for fewer epochs to show that VGER performs adequately when optimized for speed.

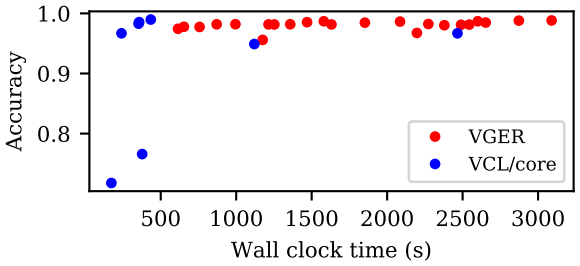

Figure 14: **Multi-headed Timed Split MNIST.** Ideal performance is in the top left corner. VGER can reach good accuracy, but slower than VCL. Now that VCL can use multi-headed training as well as coresets, the accuracy is good in the multi-headed experiment.

B.5 Timed Split MNIST Details

We perform the single-headed Split MNIST experiment with a range of different configurations. For VGER we allow between 10 and 50 epochs for training the convolutional GANs and between 1 and 120 epochs for training the main model. We use 2000, 4000, or 6000 generated images per class. We add GAN training time to the main figure, even though it is possible to do in parallel. For VCL, we use between 1 and 120 epochs for training and coresets. In all cases, we report elapsed wall-clock time from start to finish. We plot this against average accuracy over all tasks of the final model trained on all tasks. In all cases, the experiment was carried out on an NVIDIA Tesla K80.

Here, we also show Timed Split MNIST performed as a multi-headed experiment in order to allow VCL to use multi-headed training (figure 14). Performance is better for all models than in the single-headed case, but for VCL it can get very good, and faster than VGER.

B.5.1 Time and Memory efficiency of Prior- and Likelihood-focused Approaches

Likelihood-focused approaches must train generative models and train on extra data relative to prior-focused approaches. But prior-focused approaches tend have slightly more complicated losses or add extra training phases. We found that VCL with coreset was faster than VGER. For the single-headed Split MNIST task it took roughly 7 minutes.[5] Training VGER with each generated set the same size as the true data took roughly 18 minutes plus 34 minutes to train the GANs. But a quicker training regime took only 2 minutes plus 8 for training GANs and was still more accurate than VCL. [6] The prior-focused costs scale badly with model size, while VGER's costs scale badly with data-space complexity and very long series of datasets. VGER is also potentially less memory intensive. Generated data can be sampled on demand, it need not be the same each epoch.

B.6 Mutual Information MNIST

To quantify the uncertainty quality in continual learning we introduce a simple decision-rule: Fix some $1 < \theta < 10$. Then test if the uncertainty is higher than θ times the uncertainty on the current task. If so, predict that the example comes from an unseen task. We then construct an ROC plot of true positives against false positives for all θ thresholds, and compute the area under the curve (see figures 17 18). To measure uncertainty, we use the mutual information between predictions on each task and the model posterior, following Gal [4]. The mutual information is high when evaluated on data points far away from the training data.

For VGER, the obtained area under the curve of the ROC plot is 1, i.e. there is a value θ which cleanly divides high-uncertainty unseen tasks from lower uncertainty seen tasks. For VCL, the obtained area under the curve is 0.76. This is because the model stops recognizing old tasks as familiar—more specifically, the uncertainty of a model trained up until the fourth task when shown data from the second task (seen) is as high as when it is shown the fifth task (unseen). See figures 15 and 16 for full comparison.

[5] Times are reported for a Nvidia Tesla K80.

[6] Average single-headed accuracy of the final model for this less-trained VGER was 90% rather than 96% for the fully trained model. Training used one third the generated data, only 20 epochs of training, and only a fifth the training for the GAN.

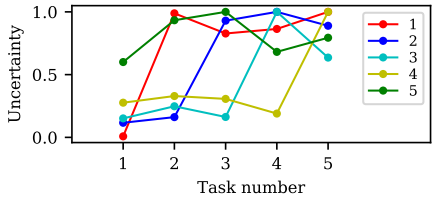

Figure 15: **Mutual Information MNIST VGER.** Uncertainty is high for unseen tasks, and low for all previously seen tasks, showing good calibration. Models trained on 1-5 tasks are each evaluated against all five tasks. Uncertainty is assessed using the mutual information between predictions on each task and the model posterior. It is normalized to 1 for the most uncertain task for each model.

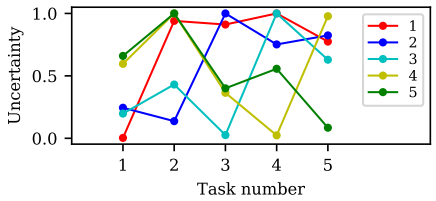

Figure 16: **Mutual Information MNIST VCL.** Uncertainty is high for unseen tasks, but rises for seen tasks that were seen some time ago. The fourth model, for example, is nearly as uncertain about the second task, which it has seen, as it is about the fifth, which it has not.

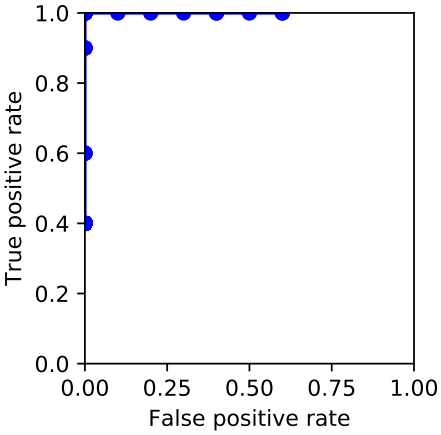

Figure 17: For VGER, sensible cut-off points cleanly differentiate previously-seen tasks from unseen ones with no false positives. This results in perfect recognition of previously seen tasks.

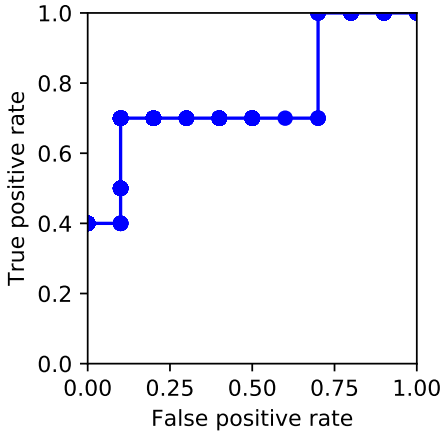

Figure 18: For VCL, false positives are a problem because the model is highly uncertain about many previously-seen tasks. This represents forgetting of old tasks.

In more detail, to assess the uncertainty of our models we compute the mutual information between predictions made on data from each task and the model posterior. This is high when the model is uncertain about predictions overall, but makes confident predictions when sampled. It is lower when the model is consistently uncertain and unconfident, and when the model is confidently correct. We measure uncertainty using exactly the same set-up as single-headed Split MNIST.

In order to measure the false positive and true positive rates, we set a decision rule that if the uncertainty on a task exceeds the uncertainty on the current task by a large enough margin, we deduce that it is a previously unseen task. We vary that margin from $\theta = 1$ (if the uncertainty is at all larger, the data must be unseen) to $\theta = 10$ (the uncertainty can be 10 times larger and we still conclude the task is one we have previously seen) with a step-size of 0.1. We then calculate the area under the curve of an ROC plot. 0.5 represents a completely worthless test, while 1 is optimal.