Combining Variational Continual Learning with FiLM Layers

Noel Loo [1] **Siddharth Swaroop** [1] **Richard E. Turner** [1]

Abstract

The standard architecture for continual learning is a multi-headed neural network, which has shared body parameters and task-specific heads. Features for each task are generated in the same way. This could be too restrictive, particularly when tasks are very distinct. We propose combining FiLM layers, a flexible way to enable task-specific feature modulation in CNNs, with an existing algorithm, Variational Continual Learning (VCL). We show that this addition consistently improves performance, particularly when tasks are more varied. Furthermore, we demonstrate how FiLM Layers can mitigate VCL's tendency to over-prune and help it use more model capacity. Finally, we find that FiLM Layers perform feature modulation as opposed to gating, making them more flexible than binary mask based approaches.

1. Introduction

In continual learning, unlike most machine learning research, datasets are not static, but rather can change in size, statistics, or number of tasks over time. This is highly applicable to real-world settings, where models are constantly re-tuned in accordance to changing demands. Standard machine learning models and training procedures fail in these settings, resulting in "catastrophic forgetting" (French, 1999), so special fitting algorithms and architectures need to be applied.

One approach to continual learning is to modify the architecture of neural networks. While it is typical for each task to have its own specific "head-network," more complex architectures add intermediate or parallel components for each task, such as in Serrà et al. (2018) or Rusu et al. (2016). These architectural approaches are motivated by the idea that model capacity should scale as tasks are added, or alternatively by the need for more task-specific differentiation beyond that provided by the "head-network."

[1]University of Cambridge, UK. Correspondence to: Noel Loo <nl355@cam.ac.uk>.

Presented at the 4th Lifelong Learning Workshop at ICML 2020.

In this paper, motivated by the inflexibility of having fully shared features and the tendency of Variational Continual Learning (VCL) to aggressively prune out hidden units, we propose combining VCL with FiLM Layers, which are flexible per-task feature modulation layers. Our method can be seen as an architecture-based approach to continual learning. In the next section, we provide background to VCL and FiLM layers. In section 3 we describe our method. We then show that it consistently improves VCL's performance in section 4. In sections 5 and 6 we demonstrate how FiLM layers mitigate overpruning and perform feature modulation rather than gating, achieving our initial motivating goals.

2. Background

Here, we detail relevant background on VCL, FiLM Layers and related continual learning approaches.

2.1. Variational Continual Learning

Variational Continual Learning (VCL) is a widely-used regularization-based approach to continual learning which works by iteratively applying Bayes' rule to form a running posterior of the model parameters given the previous datasets (Nguyen et al., 2017). At each stage, variational inference is applied to form an approximate posterior, which is then used as the prior when training the next task. When training a task, the Bayes' by Backprop algorithm (Blundell et al., 2015) is applied, optimizing the Evidence Lower Bound (ELBO) using gradient descent:

$$\text{ELBO} = \mathbb{E}_{\theta\sim q(\theta)} \log p(D_T|\theta) - D_{KL}(q(\theta)||q_{T-1}(\theta)),$$

where $q_{T-1}(\theta)$ is the previous task's approximate posterior, D_T is the task T dataset, D_{KL} is the KL-divergence, and $q(\theta)$ is the approximate posterior. VCL is model-agnostic, however the discriminative architecture used in Nguyen et al. (2017) consists of a shared set of "body" parameters, representing all the layers before the final layer, and a "head-network," with parameters specific to each task. This architecture is widely used in continual learning and multi-task learning (Kirkpatrick et al., 2016; Zenke et al., 2017; Rusu et al., 2016). The same set of features are used for each task, regardless of how similar or distinct tasks are.

2.2. FiLM Layers

While many continual learning algorithms continue to use shared features between tasks, the use of task-specific feature-wise adaptation has widely been explored in the domains of few-shot and multi-task learning (Rebuffi et al., 2017; 2018; Requeima et al., 2019). Perez et al. (2017) describes FiLM layers, which perform feature-wise modulation by applying a scale and shift to the output of convolutional layers. A FiLM layer has parameters γ and β, which respectively control the scale and shift of the transformation. In fully-connected layers, the transformation is applied element-wise: for a hidden layer with width W and activation values h_i, $1 \leq i \leq W$, FiLM layers perform the transformation $h'_i = \gamma_i h_i + \beta_i$, before this is passed on to the remainder of the network. For convolutional layers, transformations are done feature-wise. Consider a layer with N filters of size $K \times K$, resulting in activations $h_{i,j,k}$, $1 \leq i \leq N, 1 \leq j \leq W, 1 \leq k \leq H$, where W and H are the dimensions of the resulting feature map. The transformation has the form $h'_{i,j,k} = \gamma_i * h_{i,j,k} + \beta_i$. The number of required parameters scales with the number of filters, as opposed to the full activation dimension, making them computationally cheap and parameter efficient.

2.3. Related Work

There has been past work investigating per-task gating mechanisms in continual learning. Masse et al. (2018) apply a random binary mask to the nodes of a network for each task. Serrà et al. (2018) apply a similar mechanism to FiLM layers, but only apply a scale parameter, and restrict this parameter to be 0 to 1. Furthermore, a more complex training procedure involving gradient modification and parameter annealing is required. Adel et al. (2020), like us, approach per-task adaptation through variational inference, but have many parameters and is therefore relatively complex, requiring multiple subdivisions of the dataset and additional steps outside of gradient descent.

3. Variational Continual Learning with FiLM Layers

We propose combining FiLM Layers with the Variational Continual Learning algorithm. Specifically, in addition to the standard set of shared parameters θ which are fit using the VCL algorithm, we introduce a set of task-specific parameters ψ_T in the form of FiLM Layers. Unlike the body parameters, FiLM parameters are not shared between tasks so we do not need to model their uncertainties to prevent "important" parameters from changing. Therefore, we fit these FiLM layer parameters using maximum likelihood estimation, i.e. we apply no penalty to the shifts or scales.

This results in the following loss function for task T:

$$L_T = -\mathbb{E}_{\theta\sim q(\theta)} \log p(D_T|\theta, \psi_T) + D_{KL}(q(\theta)||q_{T-1}(\theta))$$

We chose VCL as opposed to other algorithms for fitting the shared body parameters because when optimizing, to minimize the KL-divergence term, there exists a non-degenerate optimal weight and scale setting. In contrast, MAP estimation leads to degeneracy in training, as the optimal solution sets the shared weights to 0, and increases the FiLM scales infinitely. A full explanation of this phenomenon is given in supplementary material section 1. As evident by the complex training procedure used in Serrà et al. (2018), encouraging effective use of scale parameters with point-estimation is difficult and requires many arbitrary rules. Furthermore, we also believe that adding a feature-wise scale invariance to tasks is a useful inductive bias, and could aid in forwards or backwards transfer.

4. Experimental Results

We applied our proposed algorithm to three sets of benchmark tasks. We test on both difficulties of the CHASY benchmark: Easy-CHASY and Hard-CHASY. We then also test on Split MNIST. The CHASY benchmark comprises of a set of tasks specifically designed for multi-task and continual learning. It is derived from the HASYv2 dataset (Thoma, 2017), which consists of 32x32 handwritten latex characters. Easy-CHASY was designed to maximize transfer between tasks and consists of tasks with 20 classes for the first task, to 11 classes for the last. Hard-CHASY represents scenarios where tasks are very distinct, where tasks range from 18 to 10 classes. Details of this benchmark are supplied in the supplementary material section 4. Hard-CHASY is is a particularly useful test for FiLM layers, as we believe that FiLM layers would allow for more flexible task-specific features, which are more useful where features differ significantly between tasks. For the two CHASY benchmarks we used a convolutional network. For our *Split*-MNIST experiment, we used a fully-connected network, which is the standard architecture used in the continual learning literature. In addition to the standard 5 binary classification tasks for *Split*-MNIST, we add 5 more binary classification tasks by taking characters from the KMNIST dataset (Clanuwat et al., 2018). Architecture and experiment details are given in the supplementary material section 2.

Figure 1 shows that there is a consistent performance gain in all three benchmark scenarios over vanilla VCL. The gain is most significant on Hard-CHASY and the Split MNIST set of tasks. This is consistent with the hypothesis that FiLM layers are more useful when used on sets of distinct tasks, as sets of similar tasks require very similar features and therefore show little benefit from per-task modulation.

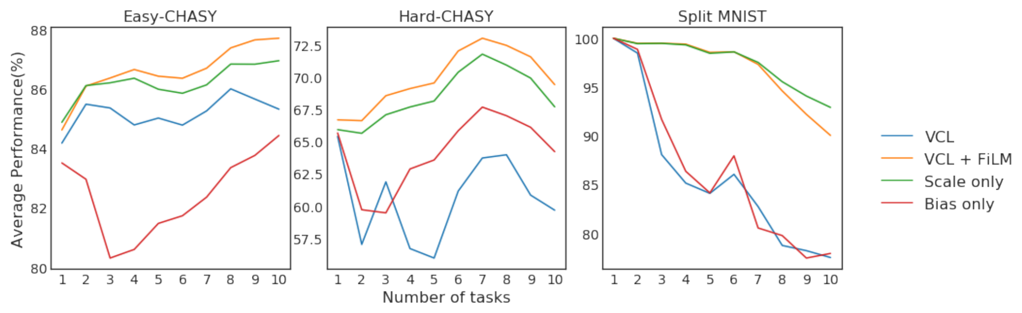


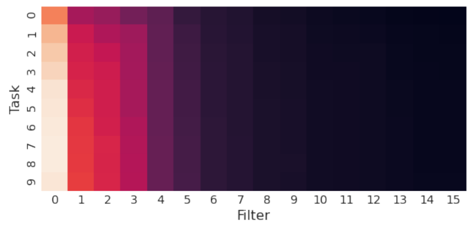
Figure 1. Performance of variational continual learning with FiLM Layers with either their biases or scales removed in comparison to standard FiLM layers and no FiLM layers, corresponding to vanilla VCL. FiLM layers consistently outperform no FiLM layers, particular when tasks are more distinct (Hard-CHASY and *Split*-MNIST tasks). The bias-only version performs similar to or worse than having no FiLM layers, while the scale-only variant performs slightly worse than standard FiLM layers. The mean over 5 runs is reported. Standard deviations are shown in figure 1 in the supplementary material.

5. FiLM Layers Mitigate Overpruning

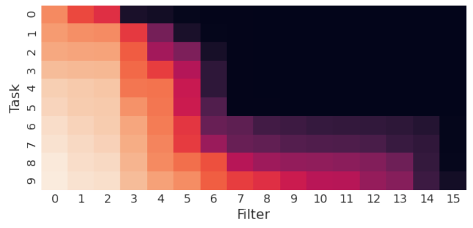
A known issue with Bayesian Neural Networks trained with variational inference is overpruning (Trippe & Turner, 2018; Turner et al., 2011), which is when a model unfits the dataset and very few nodes of the network are actually used. In the continual learning setting, this could affect performance especially when tasks are more distinct. While overpruning can free up model capacity and improve performance, as argued in Swaroop et al. (2019), it can also have a detrimental effect. Due to the overpruning, models trained continually tend not to learn new features as more tasks are trained, forcing new tasks to use existing features instead, even if they are ill-fit for the task. Figure 2a shows this phenomenon in action. As more tasks are trained sequentially in Hard-CHASY, the KL-divergence of nodes in the first convolutional layer remains constant, with the majority of nodes remaining close to the prior. This means that new features are not learnt with new tasks, going against one of the desiderata of continual learning algorithms, which is that model capacity should be allocated appropriately as more tasks are trained.

In contrast, a model fit with FiLM Layers results in figure 2b. Here, more filters begin to diverge from the prior as more tasks are trained, in line with our expectation on Hard-CHASY. To understand the cause of this, we visualised the exact posterior distributions of the weights and biases of the first layer convolutional layer after the first task was trained, in figures 3 and 4.

In figures 3a and 4a we notice that when trained without FiLM layers, the weights and biases never quite return to the prior. In particular, the bias for unused nodes concentrates at a large negative value. This causes the filter to be cut off by the ReLU activations, preventing noise from being added to the next layer. In this single task setting, this is not an issue, but when trained sequentially, subsequent tasks inherit this inactive node, and because of the shared bias being negative, no gradients can flow to the weights and the node remains inactive.



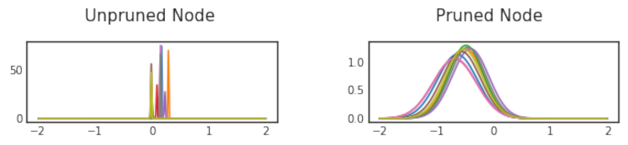
(a) Without FiLM layers



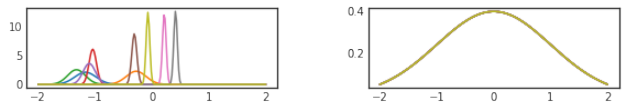
(b) With FiLM layers

Figure 2. Visualizations of deviation from the prior distribution for filters in the first layer of a convolutional networks trained on Hard-CHASY. Models are trained either sequentially using VCL (a), or sequentially with VCL + FiLM (b). FiLM layers increase the number of active units.

In contrast, because FiLM layers have unregularized shift and scale parameters, to prune a node, either the task-specific scale could be set to 0, or the shift can be set to a large negative value. This can be done without incurring any KL-divergence penalty, and the network therefore does this rather than changing the global parameters. Meanwhile, the global parameters are able to revert to the prior distribution,

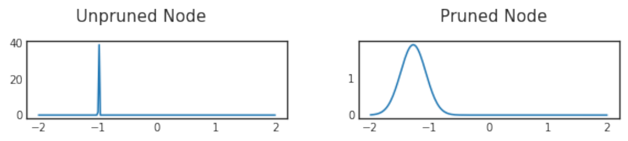


(a) Without FiLM layers

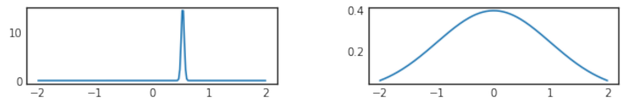


(b) With FiLM layers

Figure 3. Weight posterior distributions of pruned and unpruned nodes for the first convolutional layer after being trained on the first task in Hard-CHASY with and without FiLM Layers (figures 3a and 3b, respectively). Each line represents the posterior of an incoming weight. With FiLM Layers, unused filters have their weights return to the prior by setting the scale to zero.



(a) Without FiLM layers



(b) With FiLM layers

Figure 4. Bias posterior distributions of pruned and unpruned nodes for the first convolutional layer after being trained on the first task in Hard-CHASY with and without FiLM layers (figures 4a and 4b, respectively). In order to prune units without FiLM Layers, the shared bias must be set to a negative values, preventing node reactivation in subsequent tasks. With FiLM layers, biases in pruned units returns to the prior.

as seen by figures 3b and 4b, where unused nodes have their weights and biases set at the standard unit Gaussian prior. Now, the shared bias is no longer negative so nodes can be "reactivated" in later tasks. Furthermore, because the task-specific bias or scale performs pruning, re-activating a node in later tasks would not affect prior tasks.

6. FiLM Layers Modulate Rather than Gate

To better understand the role that FiLM Layers play in a network, we performed ablation tests on FiLM layers, either removing the scale or shift parameters. The result is shown in figure 1.

From figure 1, we notice that scale-only FiLM layers perform reasonably well, meanwhile shift-only FiLM layers perform just as poorly as having no FiLM Layers. This suggests that FiLM Layers do not perform exclusively gating. Gating can be performed using only shifts by setting unused nodes to large negative values. Feature modulation on the other hand, cannot be performed using only biases. This suggests that FiLM Layers modulate filters as well as gate them, in line with our hypothesis that feature-wise scale invariance is a useful inductive bias for continual learning. We verified this hypothesis by looking at the actual distributions of the scale parameters in FiLM Layers, as seen in figure 5.

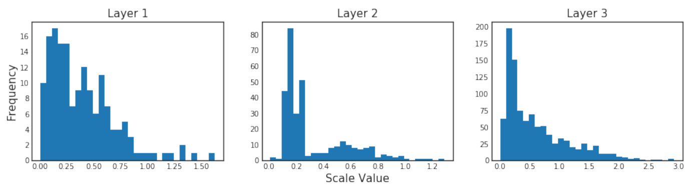


Figure 5. Distribution of the FiLM scale parameters (aggregated over all tasks) on Hard-CHASY for models trained sequentially with VCL. The scale parameters do not have a large mode at 0, and are distributed over a range of values, suggesting that the FiLM layers are not only performing gating, but actually modulating features.

If FiLM layers performed primarily a gating function, we would expect that there to be a large mode near 0, and perhaps another one at a larger value. However, the largest mode is consistently at a slightly positive value, and scales vary between 0-1.5 for the convolutional layers and 0-3 for the fully-connected layer. This continuous nature allows for more flexibility than existing approaches which rely on hard-gating, such as Masse et al. (2018). Since the scale parameters often exceeded 1, gating mechanisms which restrict scales to be between 0 and 1 such as Serrà et al. (2018) are too restrictive, and do not allow new tasks to upscale existing features from previous tasks.

7. Conclusion

In this paper we proposed the use of task-specific FiLM Layers for continual learning, motivated by the belief that all tasks having features generated by exactly the same weights is too restrictive. We combined FiLM layers with Variational Continual learning and found consistent improvements across a variety of benchmark tasks. By analyzing the effect of FiLM layers on the weight distributions of models fit with VCL, we found that FiLM layers mitigated overpruning by preventing deactivation of nodes through the shared parameters, using the task-specific parameters instead. Finally, by performing ablations on the FiLM layers and looking at the scale values of trained layers, we found that FiLM layers perform not only gating, but also modulation.

References

Adel, T., Zhao, H., and Turner, R. E. Continual learning with adaptive weights (claw). In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Hklso24Kwr.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. *ArXiv*, abs/1505.05424, 2015.

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. Deep learning for classical japanese literature. 2018. doi: 10.20676/00000341.

French, R. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3:128–135, 05 1999. doi: 10.1016/S1364-6613(99)01294-2.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks, 2016.

Masse, N. Y., Grant, G. D., and Freedman, D. J. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. 2018. doi: 10.1073/pnas.1803839115.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. Variational continual learning, 2017.

Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A. C. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2017.

Rebuffi, S.-A., Bilen, H., and Vedaldi, A. Learning multiple visual domains with residual adapters, 2017.

Rebuffi, S.-A., Vedaldi, A., and Bilen, H. Efficient parametrization of multi-domain deep neural networks. pp. 8119–8127, 06 2018. doi: 10.1109/CVPR.2018.00847.

Requeima, J., Gordon, J., Bronskill, J., Nowozin, S., and Turner, R. E. Fast and flexible multi-task classification using conditional neural adaptive processes, 2019.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks, 2016.

Serrà, J., Surís, D., Miron, M., and Karatzoglou, A. Overcoming catastrophic forgetting with hard attention to the task, 2018.

Swaroop, S., Nguyen, C. V., Bui, T. D., and Turner, R. E. Improving and understanding variational continual learning, 2019.

Thoma, M. The hasyv2 dataset, 2017.

Trippe, B. and Turner, R. Overpruning in variational bayesian neural networks, 2018.

Turner, R., Sahani, M., Barber, D., Cemgil, A., and Chiappa, S. Two problems with variational expectation maximisation for time-series models. *Bayesian Time Series Models*, pp. 109–130, 01 2011.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence, 2017.

Supplementary Material for Combining Variational Continual Learning with FiLM Layers

1 MAP Degeneracy with FiLM Layers

Here we describe how training FiLM layer with MAP training leads to degenerate values for the weights and scales, whereas with VI training, no degeneracy occurs. For simplicity, consider only the nodes leading into a single node and let there be d of them, i.e. θ has dimension d. Because we only have one node, our scale parameter γ is a single variable.

For MAP training, we have the loss function $L = -p(D|\theta, \gamma) + \frac{\lambda}{2}\theta^2$, with D the dataset and λ the L2 regularization hyperparameter. Note that $p(D|\theta, \gamma) = p(D|c\theta, \frac{1}{c}\gamma)$, hence we can scale θ arbitrarily without affecting the likelihood, so long as γ is scaled inversely. If $c < 1$, $\frac{\lambda}{2}\theta^2 < \frac{\lambda}{2}(\frac{1}{c}\theta)^2$, so increasing c decreases the L2 penalty if θ is inversely scaled by c. Therefore the optimal setting of the scale parameter γ is arbitrarily large, while θ shrinks to 0.

At a high level, VI-training (with Gaussian posteriors and priors) does not have this issue because the KL-divergence penalizes the variance of the parameters from deviating from the prior in addition to the mean parameters, whereas MAP training only penalizes the means. Unlike with MAP training, if we downscale the weights, we also downscale the value of the variances, which increases the KL-divergence. The variances cannot revert to the prior either, as when they are up-scaled by the FiLM scale parameter, the noise would increase, affecting the log-likelihood component of the ELBO. Therefore, there exists an optimal amount of scaling which balances the mean-squared penalty component of the KL-divergence and the variance terms.

Mathematically we can derive this optimal scale. Consider the scenario with VI training with Gaussian variational distribution and prior, where our approximate posterior $q(\theta)$ has mean and variance μ and Σ and our prior $p(\theta)$ has parameters μ_0 and Σ_0. First consider the scenario without FiLM Layers. Now, have our loss function $L = -\mathbb{E}_{\theta \sim q(\theta)} \log p(D|\theta) + D_{KL}(q(\theta)||q_0(\theta))$. For multivariate Gaussians,

$$D_{KL}(q(\theta)||p(\theta)) = \frac{1}{2}(\log|\Sigma_0| - \log|\Sigma| - d + Tr(\Sigma_0^{-1}\Sigma) + (\mu - \mu_0)^T\Sigma_0^{-1}(\mu - \mu_0))$$

Now consider another distribution $q'(\theta)$, with mean and variance parameters $c\mu$ and $c^2\Sigma$. Now if $q'(\theta)$ is paired with FiLM scale parameter γ set at $\frac{1}{c}$, the log-likelihood component is unchanged:

$$\mathbb{E}_{\theta \sim q(\theta)} \log p(D|\theta) = \mathbb{E}_{\theta \sim q'(\theta)} \log p(D|\theta, \gamma = \frac{1}{c})$$

with γ being our FiLM scale parameter and $p(D|\theta, \gamma)$ representing a model with FiLM scale layers. Now consider the $D_{KL}(q'(\theta)||q_0(\theta))$, and optimize c with μ and Σ fixed:

$$\begin{aligned}
D_{KL}(q'(\theta)||p(\theta)) &= \frac{1}{2}(\log|\Sigma_0| - \log|c^2\Sigma| - d + Tr(\Sigma_0^{-1}c^2\Sigma) + (c\mu-\mu_0)^T\Sigma_0^{-1}(c\mu-\mu_0)) \\
&= \frac{1}{2}(\log|\Sigma_0| - \log|\Sigma| - 2d\log c - d + c^2Tr(\Sigma_0^{-1}\Sigma) + (c\mu-\mu_0)^T\Sigma_0^{-1}(c\mu-\mu_0)) \\
\frac{\partial D_{KL}}{\partial c}|_{c=c^*} = 0 &= -\frac{d}{c^*} + c^*Tr(\Sigma_0^{-1}\Sigma) + (c^*\mu-\mu_0)^T\Sigma_0^{-1}\mu \\
0 &= -d + c^{*2}Tr(\Sigma_0^{-1}\Sigma) + c^{*2}\mu^T\Sigma_0^{-1}\mu - c^*\mu_0^T\Sigma_0^{-1}\mu \\
0 &= c^{*2}(Tr(\Sigma_0^{-1}\Sigma) + \mu^T\Sigma_0^{-1}\mu) - c^*\mu_0^T\Sigma_0^{-1}\mu - d \\
\Rightarrow c^* &= \frac{\mu_0^T\Sigma_0^{-1}\mu \pm \sqrt{(\mu_0^T\Sigma_0^{-1}\mu)^2 + 4d(Tr(\Sigma_0^{-1}\Sigma) + \mu^T\Sigma_0^{-1}\mu)}}{2(Tr(\Sigma_0^{-1}\Sigma) + \mu^T\Sigma_0^{-1}\mu)}
\end{aligned}$$

Also note that $c = 0$ results in an infinitely-large KL-divergence, so there is a barrier at $c = 0$, i.e. If optimized through gradient descent, c should never change sign. Furthermore, note that

$$\frac{\partial^2 D_{KL}}{\partial c^2} = \frac{d}{c^2} + Tr(\Sigma_0^{-1}\Sigma) + \mu^T\Sigma_0^{-1}\mu > 0$$

So the KL-divergence is concave with respective to c, so c^* is a minimizer of D_{KL} and therefore

$$D_{KL}(q(\theta)||p(\theta)) \geq D_{KL}(q'(\theta)||p(\theta))|_{c=c*}$$

Which implies the optimal value of the FiLM scale parameter γ is $\frac{1}{c^*}$. While no formal data was collected, it was observed that the scale parameters do in fact reach very close to this optimal scale value after training.

2 Model and Training Parameters

All models trained on images from the HASYv2 dataset used the same convolutional model. This model consists of: 16-wide 3x3 convolutional layer, 2x2 max pooling, ReLU, 32-wide 3x3 convolutional layer, 2x2 max pooling, ReLU, flattening, 100-wide fully connected layer, then ReLU. The final head layer size is determined by the number of classes in the task.

Split-MNIST models use a 2-layer MLP with 256 hidden units per layer and ReLU activations. Unless otherwise stated, the batch size for training was 64 and the learning rate was 0.001.

Unless otherwise stated, all results are averaged over 5 repeats of the experiment with 5 different random seeds. In the case of the HASYv2 dataset, there were 5 different train/test splits which were common across all experiments dealing with that dataset. The means of these five runs are reported.

FiLM scale and biases are initialized at 1 and 0, respectively.

2.1 Continual Learning Experiments

Models trained with variational continual learning were trained for 1000 epochs for HASYv2 tasks and 50 for split-MNIST. With no early stopping. The local reparameterization trick (Kingma, Salimans, and Welling 2015) was used in accordance to the suggestions in (Swaroop et al. 2019).

For Figures 2 and 3, 4, we looked at a network trained using a slightly modified version of VCL where the KL-divergence term was down-weighted by a factor of 5. This was necessary, as without it many nodes failed to converge in the allotted epochs, making each task have a larger KL-divergence from the prior for all nodes, making it difficult to discern which nodes were "active".

3 Additional Plots

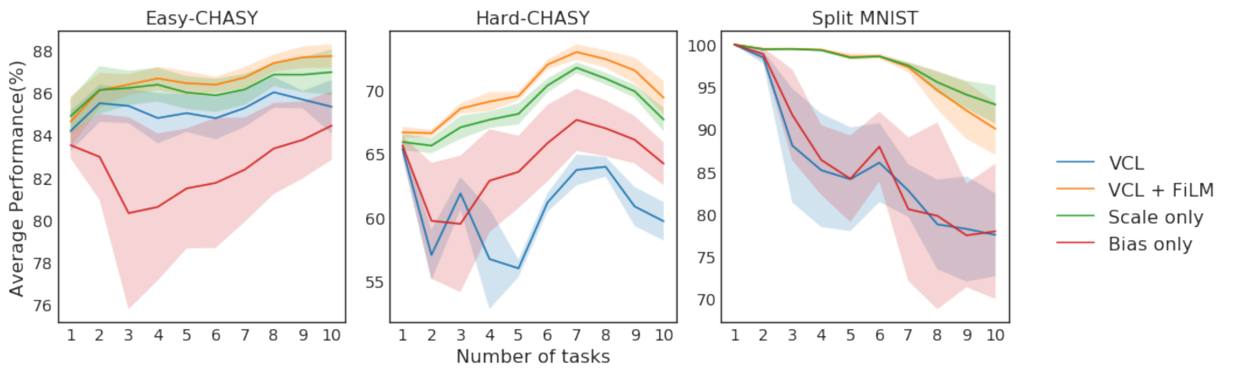
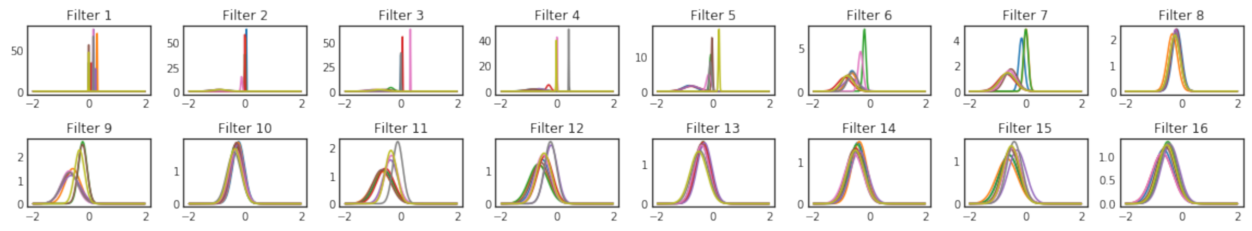
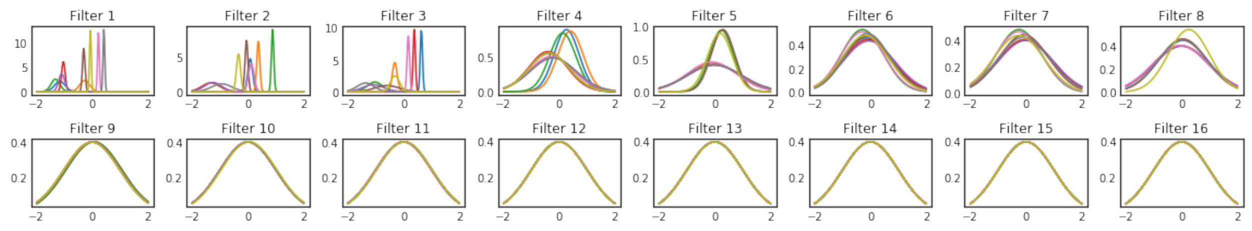


Figure 1: Performance of variational continual learning with FiLM Layers with either their biases or scales removed in comparison to standard FiLM layers and no FiLM layers, corresponding to vanilla VCL. Standard deviations are given by in the shaded region. FiLM layers consistently outperform no FiLM layers, particular when tasks are more distinct (Hard-CHASY and *Split*-MNIST tasks). The bias-only version performs similar to or worse than having no FiLM layers, while the scale-only variant performs slightly worse than standard FiLM layers. Each line representats the posterior distribution of a weight in the filter.

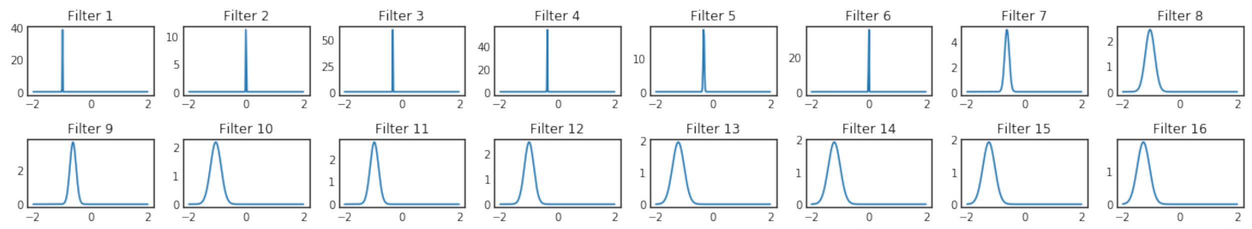


(a) Without FiLM layers

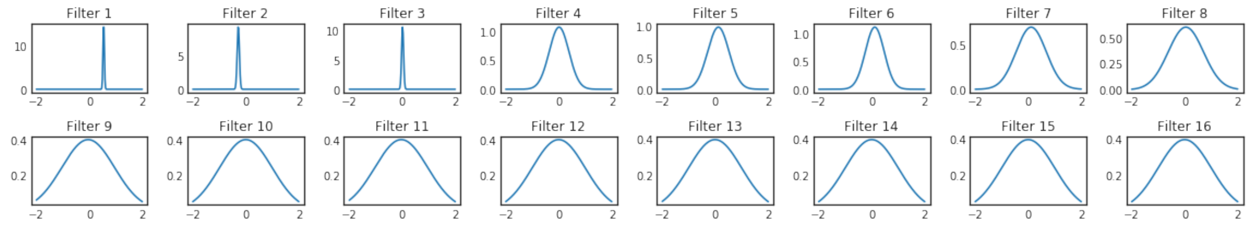


(b) With FiLM layers

Figure 2: Weight posterior distributions of incoming weights for the first convolutional layer after being trained on the first task in Hard-CHASY with no FiLM Layers (figure 2a) and with FiLM Layers (figure 2b). With FiLM Layers, unused filters can have their weights return to their prior by setting the scale to zero, as seen by filters 9-16 in figure 2b



(a) Without FiLM layers



(b) With FiLM layers

Figure 3: Bias posterior distributions of incoming biases for the first convolutional layer after being trained on the first task in Hard-CHASY with no FiLM Layers (figure 3a) and with FiLM Layers (figure 3b). In order to prune units without FiLM Layers, the shared bias must be set to a negative values, preventing node reactivation in subsequent tasks. The bias in pruned units returns to the prior when FiLM layers are used as seen by filters 9-16 in figure 3b

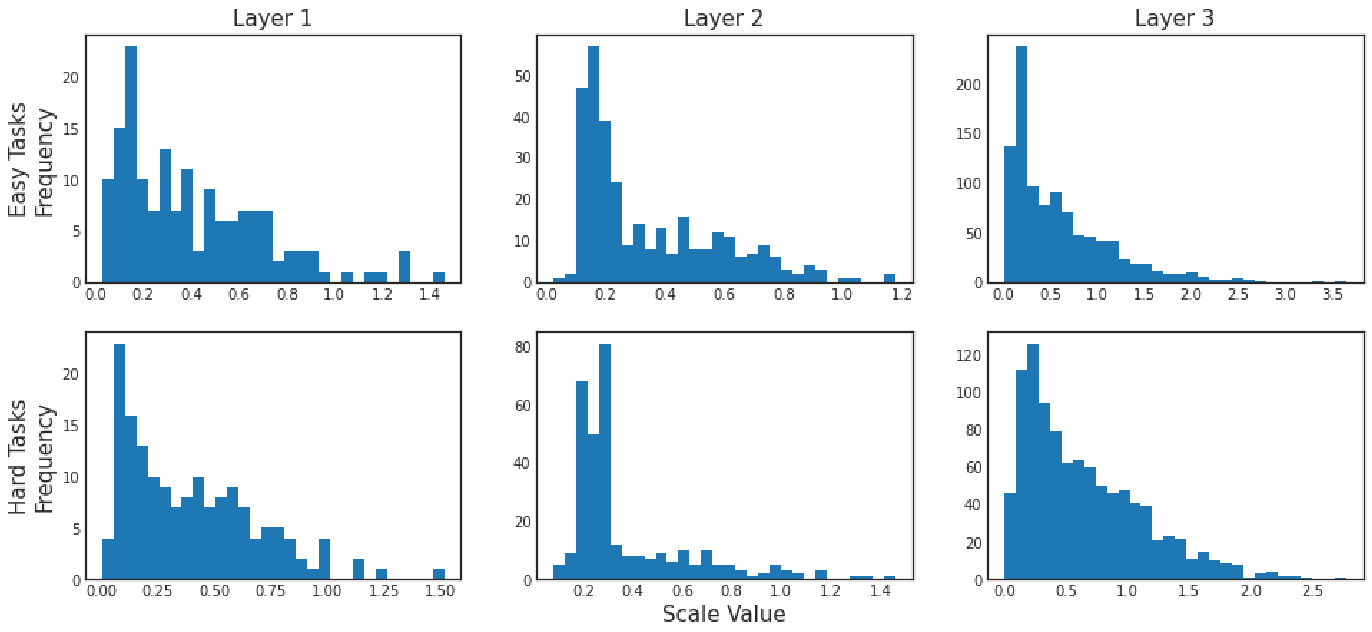


Figure 4: Distribution of the FiLM scale parameters (aggregated over all tasks) on Easy-CHASY (top row) and Hard-CHASY (bottom row) for models trained sequentially with VCL. The scale parameters do not have a large mode at 0, and are distributed over a range of values, suggesting that the FiLM layers are not only performing gating, but actually modulating features.

4 CHASY benchmark details

As discussed in the main text, we used the Clustered HASYv2 (CHASY) benchmark for our continual learning tests. This dataset was first proposed in Noel Loo's masters thesis, and has not yet been published. Hence, we next present the relevant section of *Anonymous*. Further information about this benchmark will be published online soon.

4.1 Clustered HASYv2 (CHASY)

The HASYv2 dataset is a dataset consisting over 32x32 black/white handwritten Latex characters. There are a total of 369 classes, and over 150 000 total samples (Thoma 2017).

We constructed 10 classification tasks, each with a varying number of classes ranging from 20 to 11. To construct these tasks, we first trained a mean-field Bayesian neural network on a 200-way classification task on the 200 classes with the most total samples. To get an embedding for each class, we use the activations of the second-last layer. Then, we performed K-means clustering with 20 clusters on the means of the embedding generated by each class when the samples of the classes were input into the network. Doing this yielded the classes shown in figure 5. Now, within each cluster are classes which are deemed "similar" by the network. To make the 10 classification tasks, we then took classes from each cluster sequentially (in order of the class whose mean was closest to the cluster's mean), so that each task contains at most 1 symbol from each cluster. Doing this ensures that tasks are similar to one another, since each task consists of classes which are different in similar ways. With the classes selected, the training set is made by selecting 16 samples of each classes, and using the remaining as the test set. This procedure was used to generate the "easy" set of tasks, which should have the maximum amount of similarity between tasks. We also constructed a second set of tasks, the "hard" set, in which each task is individually difficult. This was done by selecting each task to be classification within each cluster, selecting clusters with the most number of symbols first. This corresponds to clusters 3, 5, 10, 0, 12, 7, 9, 13, 19 and 8 in figure 5. With the classes for each task selected, 16 samples from each class are used in the training set, and the remainder are used as the test set. Excess samples are discarded so that the test set class distribution is also uniform within each task.

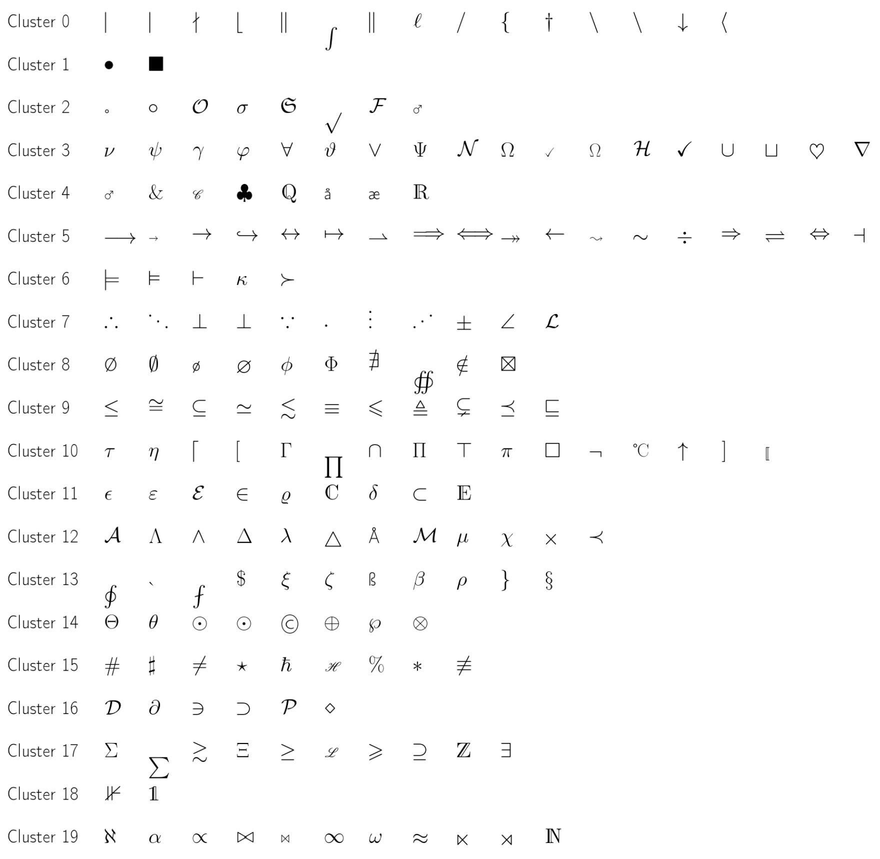
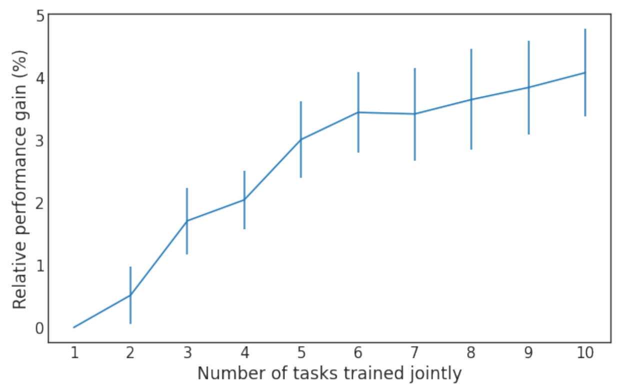


Figure 5: Clusters of symbols found by performing K-means clustering with $K = 20$ based on the embedding layer of a model trained with variational inference on a 200-way classification task on the 200 most common symbols in the HASYv2 dataset. The "Easy" set of tasks is made by taking the first symbol from each cluster as the first task, then the second, and so on, up to 10 tasks. The "Hard" set of tasks in made by taking the clusters with the most classes in order (3, 5, 10, 0, 12, 7, 9, 13, 19, 8)

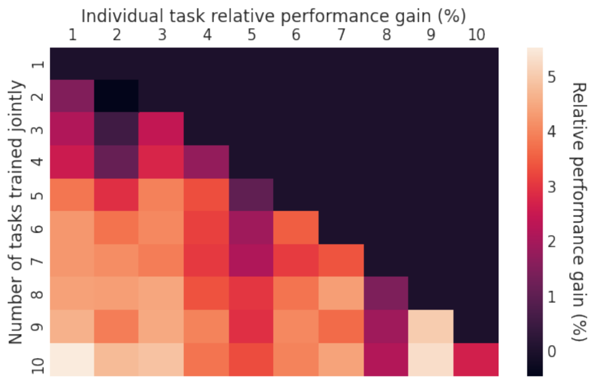
.

It was necessary to perform this clustering procedure as we found it difficult to produce sizable transfer gains if we simply constructed tasks by taking the classes with the most samples. While we were able to have gains of up to 3% from joint training on 10 20-way classification tasks with the tasks chosen by class sample count, these gains were significantly diminished when performing MAP estimation as opposed to MLE estimation, and reduced even further when performing VI. Because one of our benchmark continual learning methods is VCL, showing transfer when trained using VI is necessary.

Figures 6a and 7 show the performance gains of joint training over separate training on this new dataset, for both MAP, and KL-reweighted VI, respectively. Figure 6b shows how relative test set accuracy varies for each specific task for these training procedures.



(a) Average relative performance



(b) Individual task relative performances

Figure 6: Relative test-set accuracy of models trained jointly on the easy set of tasks relative to individual training for MAP estimation. Figure 6a shows the means aggregated over all tasks while figure 6b shows the performance differences for individual tasks. Performance increases near monotonically as more tasks are added, achieving an average of around 4.5% gain with 10 tasks

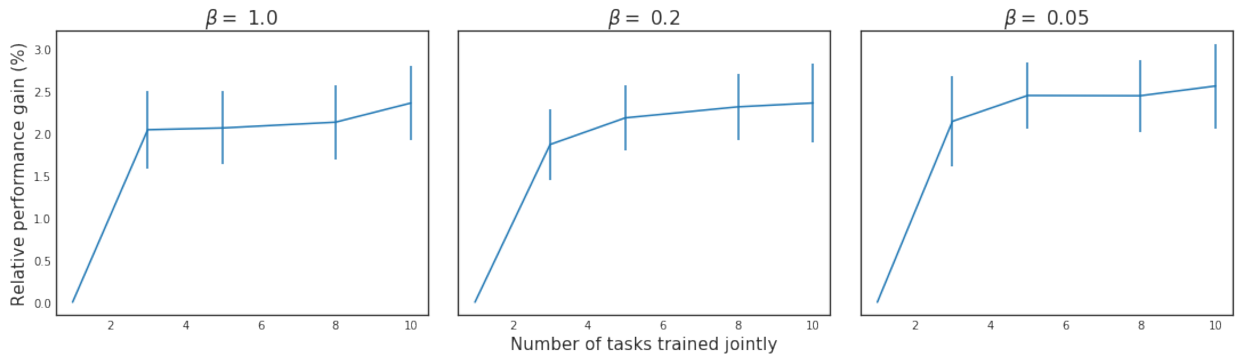


Figure 7: Relative performance of models trained jointly on the easy set of tasks relative to individual training for variational inference with various KL-reweighting coefficients β. Performance gains reach around 2.5% with 10 tasks, which is less than with MAP training but still significant

References

[1] Diederik P. Kingma, Tim Salimans, and Max Welling. *Variational Dropout and the Local Reparameterization Trick*. 2015. eprint: `arXiv:1506.02557`.

[2] Siddharth Swaroop et al. *Improving and Understanding Variational Continual Learning*. 2019. eprint: `arXiv:1905.02099`.

[3] Martin Thoma. *The HASYv2 dataset*. 2017. eprint: `arXiv:1701.08380`.