Structuring parameter uncertainity for efficient incremental learning

Vishal Keshav
University of Massachusetts Amherst
Amherst, MA 01002
vkeshav@cs.umass.edu

Abstract

Continual learning methods aim to learn a set of related tasks arriving sequentially while avoiding the catastrophic forgetting on the older tasks. These approaches are parameter and computation heavy as they need to track additional parameters that incorporate the knowledge of which part of the network needs to be preserved and which part can be further adapted. These methods become impractical on highly resource-constrained embedded devices where data is continuously generated and consumed, and where privacy and latency is the top priority. In this paper, we exploit the structural and computational properties of a class of neural networks, specifically the convolution neural network (CNN), and propose to reduce the parameter complexity without degrading the model performance. Our method is based on a regularization-based continual learning method that uses the Bayesian framework to learn group parameter uncertainty and use it as a proxy to adapt the network on new tasks. The experimental evidence validates the correctness of our *structural adaptability bias* hypothesis and better storage complexity as compared to the state-of-the-art regularization based methods.

1 Introduction

Life-long learning or continual learning [1] [2] [3] is a paradigm of machine learning where model parameters are incrementally adapted to incorporate new knowledge as data distribution changes over time (also termed as concept drift [4]). The change in the data distribution can be abrupt, for instance, the addition of a new class in the classification problem, or it can be gradual that best reflects the evolution of an environment, for example in reinforcement learning problems. Based on how data evolves, continual learning is categorized into a) Task incremental learning[5], b) Domain incremental learning[6] and c) Class incremental learning[7]. The focus of this paper remains at the class incremental learning where task identity is known during test time. Regardless of these classifications, incremental learning is prone to catastrophic forgetting[8][9][10], a phenomenon where training on new data/class deteriorates the performance of the model on old data/class. There are three main class of methods to deal with catastrophic forgetting problem, namely, dynamic architecture methods[11][12][13], memory-based methods[9][14][15] and regularization-based methods[16][17][18][19]. While dynamic architecture methods become computationally heavy as the model evolves with the data, memory-based methods require additional storage to store the past data or some form of past representative data. These two method classes are specifically troublesome in a scenario where both storage and compute is limited[20], such as resource-constrained embedded devices. Regularization-based methods, on the other hand, propose to adapt the model parameters without increasing the model complexity or storing any additional training data. In this paper, we propose to reduce the storage and computational complexity without degrading the model performance, hence we must adhere to the regularization-based approach.

Regularization-based methods modify the learning objective that suppresses the changes in the *important* parameters to avoid forgetting of the previously learned tasks. The notion of importance differs from one method to another. The recent state-of-the-art methods use Bayesian framework[21][22][23] to capture the parameter uncertainty and uses it as a proxy for the importance. The lower the parameter uncertainty, the higher the importance, and hence these parameters are less likely to change when the new data arrives. While these methods are generic enough to apply to a variety of models, not much investigation has been done to incorporate the structural assumptions of the neural network models to improve storage and computational efficiency. In this paper, we investigate how to structure the parameter uncertainty (and hence the adaptability) that can result in a practical gain in training efficiency, reduces the number of additional adaptability parameters, and improve the model performance. Specifically, we turn towards vision architectures that include convolution operators and exploit the inherent structure present in convolution operations to propose how to group the parameters that can share same uncertainty and reflects the adaptability bias for new data or task. We call our hypothesis of structurally adapting parameters in the group as *structured adaptability bias* hypothesis.

We enumerate the main contributions of this paper below:

- We study the state-of-the-art Bayesian approach that deals with learning the parameter adaptability in the context of computer vision tasks such as image classification.
- We exploit the network structure(convolution architectures) and impose an inductive bias to improve storage complexity without degrading the model performance.
- With our experiments on three visual data-sets, we empirically demonstrate the correctness of *structured adaptability bias* for CNN architectures.

Novelty Statement

We exploit the computational structure of convolutions and re-parametrize the uncertainty of the weights in the Bayesian framework that serves as a direct proxy for parameter adaptation in the context of incremental learning. The proposed learning algorithm adapts the learning rate for a group of the parameters at once, thus, allowing a gain in storage and practical computational efficiency without any degradation in model performance.

2 Related Work

In this section, we discuss the regularization-based approaches, provide background on the Bayesian framework, followed by our review of Bayesian approaches on which we build our work.

2.1 Regularization-based incremental learning

The regularization-based approach controls the catastrophic forgetting by constraining the updates to the parameters of a model. To do that, additional regularization terms are added to the optimization objective such that any updates that lead to forgetting are penalized. One of the earlier approach called Learning without forgetting (LWF) [16] used knowledge distillation as a means to retain knowledge from all the previous tasks. This method fails when the data from the new task has a drastically different distribution than what was seen in the previous tasks.

Another influential method called Elastic Weight Consolidation (EWC) [24][25] introduced the notion of parameter importance and used it to regularize the parameter updates when network is trained on new task. For two sequential tasks A and B with corresponding data $\mathcal{D}_A$ and $\mathcal{D}_B$ respectively, EWC models the parameter posterior distribution $P(w|\mathcal{D}_B, \mathcal{D}_A)$ as shown in eq. 1:

$$P(w|\mathcal{D}_B, \mathcal{D}_A) = \frac{P(\mathcal{D}_B|\mathcal{D}_A, w) * P(\mathcal{D}_A|w)}{P(\mathcal{D}_B)} \tag{1}$$

The posterior $P(\mathcal{D}_A|w)$ becomes the prior for the subsequent task B. This framework extends to T sequential tasks. Since the true posteriors are intractable, EWC estimates the distribution using Laplace approximation with precision determined by the Fisher Information matrix. Though our method uses the Bayesian framework, we do not model the problem based on eq. 1.

2.2 Bayesian framework for continual learning

Consider a discriminative neural network model, $p(y|x, w)$, that produces a probability distribution over the output y given an input x and parameters w. In Bayesian setting, w is assumed to follow a prior distribution denoted by $p(w)$ and a posterior distribution denoted by $p(w|\mathcal{D})$. The exact estimation of the posterior is intractable for a sufficiency large neural network models. To approximate this, methods such as Markov Chain Monte Carlo samples are used. These sampling methods, however, are expensive. In variational methods, the posterior is approximated by assuming a simple tractable distribution such as Gaussian for the posterior.

2.2.1 Variational Inference

The variational inference methods finds a variational distribution that approximate the posterior distribution on the weights of the neural network $P(w|\mathcal{D})$ by minimizing the Kullback-Leibler distance between assumed variational distribution $q(w|\theta)$ and $P(w|\mathcal{D})$. The maximization results to an optimal parameter θ^* that approximate $P(w|\mathcal{D})$. Formally,

$$\theta^* = \arg\min_{\theta} KL[q(w|\theta)||P(w|\mathcal{D})] \tag{2}$$

$$= \arg\min_{\theta} KL[q(w|\theta)||P(w)] - E_{q(w|\theta)}[logP(\mathcal{D}|w)] \tag{3}$$

The right side of the equation is expected lower bound(ELBO). The parameter θ can be seen as a factor that compresses the observed data $\mathcal{D}$. The minimization objective is given by $\mathcal{L}(\mathcal{D}, \theta)$:

$$\mathcal{L}(\mathcal{D}, \theta) = KL[q(w|\theta)||P(w)] - E_{q(w|\theta)}[logP(\mathcal{D}|w)] \tag{4}$$

2.2.2 Bayes by Backprop

Under certain conditions, the gradients for the objective $\mathcal{L}(\mathcal{D}, \theta)$ can be estimated by using Monte Carlo samples. Specifically, Blundell et. al. [26] shows that with reparametrization trick, the Monte Carlo samples of w can be used to estimate the unbiased estimates of $\nabla_\theta \mathcal{L}(\mathcal{D}, \theta)$. This technique enables us to directly use back-propagation to learn the variational parameters θ and is called Bayes by Backprop. If the posteriors is approximated by a Gaussian distribution $\mathcal{N}(w|\mu, \sigma^2)$, then Bayes by Backprop follows these steps to learn the parameter θ:

1. Sample ϵ from standard normal distribution N times.
2. Compute the N samples of w using re-parametrization trick, $w = \mu + \sigma * \epsilon$.
3. Compute the objective function from the samples of w,
   $\mathcal{L}(\mathcal{D}, \mu, \sigma) = \sum_{i=1}^{N} log(q(w^{(i)}|\mu, \sigma)) - logP(w^{(i)}) - logP(\mathcal{D}|w^{(i)})$.
4. Calculate the gradients $\nabla_\mu \mathcal{L}(\mathcal{D}, \mu, \sigma)$ and $\nabla_\sigma \mathcal{L}(\mathcal{D}, \mu, \sigma)$ using back-propagation.

Bayes by Backprop is applicable to the minibatch training, hence, this method can be used to train large scale neural network models, such as convolution neural network.

Blundell et. al. [26] also proposes to use the scaled mixture model of diagonal Gaussian for the prior $P(\theta)$. In this paper, we also use a fixed scaled mixture model for prior instead of posterior over the previous task.

2.3 State-of-the-art approaches

State of the art regularization-based continual learning approaches[27][22][28] uses Bayesian learning framework where they approximate the weight posterior using variational distribution. UCB[27] uses Bayes by Backprop to to update the parameters of the variational posterior for every new target tasks. Once the model is learned on a task t, the uncertainty parameters σ is used to update the learning rate for the associated parameter μ of the posterior distribution $q(w|\mu, \sigma)$ for the subsequent task $t + 1$. Let the learning rate for μ and σ is denoted by α_μ and α_σ respectively, then for each subsequent task $t + 1$, the learning rate update α_μ^{t+1} is given by eq. 5.

$$\alpha_\mu^{t+1} = \alpha_\mu^t * \sigma^t \tag{5}$$

where σ^t is the standard deviation in w learned after completing the training on task t and α_μ^t is the learning rate used for parameters μ during training on task t. Learning rate for σ is kept fixed.

Informally, if the weight variable w has a large uncertainty associated with it, then the update to the parameter μ will be comparatively large for the next task $t+1$ as compared to task t. This method, however, doubles the number of parameters for every weight in the neural network. In our approach, we relax the condition to compute per-parameter uncertainty for a given class of architecture and show that an equivalent accuracy can be obtained if the structure of the network architecture is exploited.

Another state-of-the-art method CLAW [28] also uses the Bayesian framework, but unlike ours, their approach fall in the Variational Continual Learning framework[23] framework. We fix our prior to the scaled mixture of Gaussians as described earlier. They also demonstrate that when uncertainty parameters are shared in *nodes*(a group of activation), the total number of parameters can be reduced. Unlike theirs, our method follows UCB [27] where we share the uncertainty among weights instead of activation. UCB [27] does not investigate how uncertainty affects a group of parameters, which is the main focus of our work. We explore and exploit the convolution architecture and show that with a significantly lower number of uncertainty parameters, a robust model can be trained.

Other recent work that investigate CNN architectures for continual learning are [29][30]. However, their learning approach does not fall into the Bayesian learning framework, hence, a deviation to what is presented in this paper.

3 Methodology

In this section, we first describe the *structured adaptability bias* hypothesis and use this to propose the grouping of parameters that exploit the structure for convolution architectures.

3.1 Structured adaptability bias hypothesis

In a continual learning setting, when a model is trained on a sequence of T tasks, the global parts of the model are slowly adapted to perform a global task and task-specific model parameters are quickly adapted to perform local tasks. This idea has been explored in [31][32][33]. The objective is to stabilize the adaptation of global parameters that are common to all T tasks and regularize the free parameters that reflect the properties of the local task. We hypothesize that when the parameters are adapted in groups by exploiting how the parameters are structured in a neural network, this can lead to stability in the learning of the global features of the network. A group of parameters can be seen as learning-related concepts, and when we regularize the group at once, stability in learning and reduction in catastrophic forgetting can be achieved. It has to be noted that there are no group adaptation constraints on local parts of the model, however, to improve the storage complexity, we group the local parts as well. Concretely, for convolution architectures, when we group the adaptability of parameters at the level of channel, filter, or layer, we stabilize the learning of global features.

The structural adaptability bias for CNN architectures can be argued by visualizing the filter maps and noting that each channel, filter, or layer captures some statistical relationships associated with performing a common task at a different level of abstraction[34].

3.2 Approach

Let $w_{fcij}^{(l)}$ be the weight parameter of a convolutional neural network present at layer l, filter f, channel c and at the spatial coordinate of (i, j) where $l \in \{1, 2, ..., L\}$, $f \in \{1, 2, ..., F_l\}$, $c \in \{1, 2, ..., C_f^{(l)}\}$, $i \in \{1, 2, ..., M_{fc}^{(l)}\}$ and $j \in \{1, 2, ..., N_{fc}^{(l)}\}$. L denotes the number of layers in the CNN architecture, F_l denotes the number of filters at layer l, $C_f^{(l)}$ denotes the number of channels in f^{th} filter of l^{th} layer, $M_{fc}^{(l)}$ and $N_{fc}^{(l)}$ denotes the number of parameters in spatial dimension of a channel c of filter f at layer l. UCB models the posterior over w using a Gaussian with diagonal co-varaince. For CNN architectures (assuming no bias parameters), this translates to

$q(w|\mu,\sigma) = \mathcal{N}(w|\mu,\Sigma)$ or $w^{(l)}_{fcij} \sim \mathcal{N}(w|\mu^{(l)}_{fcij}, \sigma^{(l)}_{fcij})$ where Σ is a diagonal matrix.

We propose to model the w where Σ is a block diagonal matrix and each block represents the parameters in a channel, stacked group of channels, filters or layers. Specifically, when the blocks represents channel of the filter in CNN, then $w^{(l)}_{fcij} \sim \mathcal{N}(w|\mu^{(l)}_{fcij}, \sigma^{(l)}_{fc})\ \forall l, f, c, i, j$. When the block represents a group of channels (similar to group convolutions), $w^{(l)}_{fcij} \sim \mathcal{N}(w|\mu^{(l)}_{fcij}, \sigma^{(l)}_{fg})$ $\forall l, f, c, i, j$ where g denotes the group to which the channel belongs to in a particular filter.

This modelling can be extended to filter and layers. When the block represents a filter, then $w^{(l)}_{fcij} \sim \mathcal{N}(w|\mu^{(l)}_{fcij}, \sigma^{(l)}_{f})\ \forall l, f, c, i, j$. Moreover, when the block represents a layer, then $w^{(l)}_{fcij} \sim \mathcal{N}(w|\mu^{(l)}_{fcij}, \sigma^{(l)})\ \forall l, f, c, i, j$. The last case of parameter grouping is particularly interesting because this directly interprets to dynamically freezing the layers of neural network for knowledge transfer[35] from task t to $t+1$. We refer the proposed four approaches of modelling posterior on w as **channel-wise**, **group-channel-wise**, **filter-wise** and **layer-wise** respectively. Fig. 1 demonstrate the approaches.

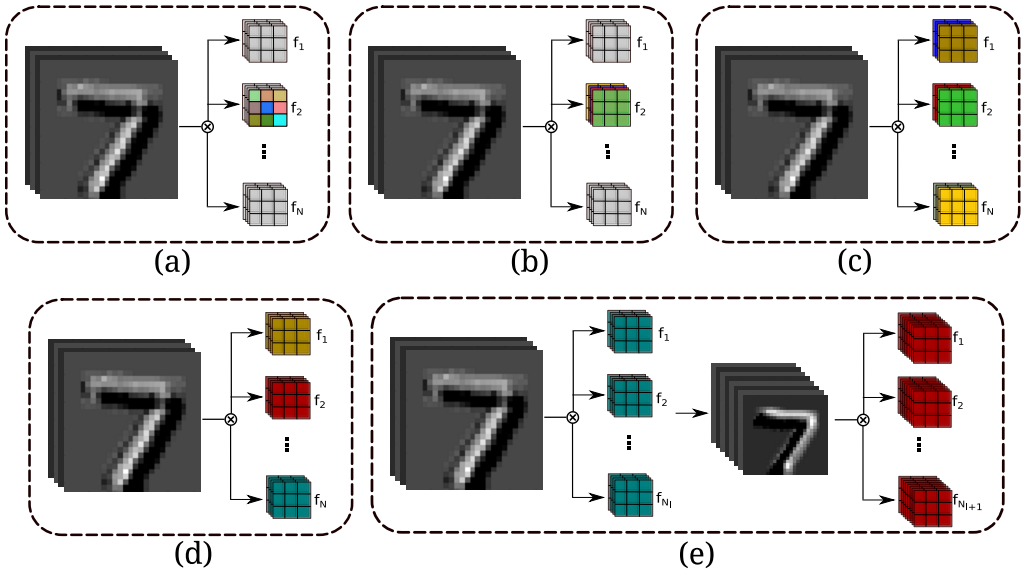


Figure 1: We use different colors to demonstrate different variances associated with the parameters. Due to the lack of many colors, we use grey color for case (a) and (b) to show the repetition of colored filtered. (a) The baseline approach used by UCB. Each parameter at each filter has its own uncertainty. (b) channel-wise, uncertainty parameters are shared by each channel of a filter. (c) group-channel-wise, uncertainty parameters are shared by a group of channels per-filter. The group size shown in the figure is set to 2. (d) filter-wise, uncertainty parameter is shared by each individual filter in a layer. (e) layer-wise, each layer has a single uncertainty parameter.

3.3 Learning Algorithm

Based on the modeling assumption proposed in the previous section, the learning algorithm that combines the Bayesian learning to learning shared uncertainty and adapt the weight parameters is presented in Alg.1. b refers to either of the four approaches to construct a block of parameters presented in Fig.1.

Algorithm 1: Training algorithm.

Input: A sequence of T data sets $\mathcal{D}_t = \{x_t^{(i)}, y_t^{(i)}\}_{i=1}^{N_t}$ where N_t is the size of data set associated with task t.

Output: $\mu_{fcij}^{(l)}$, the mean parameters of the posterior distribution

$\alpha_\mu = \alpha_\rho = \alpha$; ▷ *Initialization of the learning rate for μ and unconstrained ρ parameters*

for *t=1 to T* **do**

  while *Loss does not converge to zero* **do**

    $\epsilon \sim \mathcal{N}(0, I)$; ▷ *Sample N samples from standard normal*

    $\sigma_b = \log(1 + exp(\rho_b))$; ▷ *Constrain the variances for a group of parameters*

    $w = \mu + \sigma_b * \epsilon$; ▷ *Re-parametrization trick*

    $\mathcal{L}(\mathcal{D}_t, \mu, \rho) = \sum_{i=1}^{N}(\log(\mathcal{N}(w_i|\mu, \sigma_b)) - \log(P(w_i)) + \sum_{j=1}^{|\mathcal{D}_t|}\log(P(y_t^{(j)}|x_t^{(j)}, w_i)))$

    $\mu = \mu - \alpha_\mu \nabla_\mu \mathcal{L}(\mathcal{D}_t, \mu, \rho)$, $\rho = \rho - \alpha_\rho \nabla_\rho \mathcal{L}(\mathcal{D}_t, \mu, \rho)$; ▷ *Gradient update*

  end

  $\alpha_\mu = \alpha_\mu \sigma_b$; ▷ *Update the learning rate for the group of parameters*

end

4 Data Sets

We evaluate our approach on three data sets.

5-Split MNIST: The 10 class MNIST[36] data set is divided into 5 datasets with 2 classes each. Each of the 5 data sets serves as 5 different tasks in the context of task incremental learning.

5-Split FashionMNIST: Similar to MNIST, FashionMNIST[37] is a visual data set having 10 classes, where we split it into 5 tasks with 2 class each.

Permuted MNIST: To better understand the implication of sharing the learning rates among a group of parameters, we experiment with MNIST data set where we generate 10 tasks by applying 10 different permutations on the image pixels. This data set has been used in [19], however, we use it to justify that if global properties are not present across different tasks, the group adaptation does not work. This presents an empirical justification for *structured adaptability bias* hypothesis.

Each of the above data sets were split into $50,000$, $10,000$ and $10,000$ for train, validation and test respectively.

Table 1: Data set description.

Data-set	Input feature description	Number of tasks	Number of class per-task
5-Split MNIST	Image feature $(1 \times 28 \times 28)$	5	2
5-Split FashionMNIST	Image feature $(1 \times 28 \times 28)$	5	2
Permuted MNIST	Image feature $(1 \times 28 \times 28)$	10	10

5 Experiment

In this section, we describe the performance assessment metrics, experimental setup to compare the baseline and our approaches on three data sets, and the implementation detail.

5.1 Performance assessment metrics

Let T be the number of tasks. The average accuracy measures how well the model did after it was trained on all T tasks. Formally, average accuracy is defined in Eq. 6.

$$ACC = \frac{1}{T}\sum_{i=1}^{T} R_i^T \tag{6}$$

Backward transfer measures how the learning on task t has resulted in forgetting on tasks $\{1, 2, ..., t-1\}$. Backward transfer is defined in Eq. 7.

$$BWT = \frac{1}{T}\sum_{i=1}^{T} R_i^T - R_i^i \tag{7}$$

R_i^t is the accuracy of the model on task i after training on t sequential tasks. If BWT is positive, that means the training on new task has improved the model performance on the old task. If it is negative, this means training on new task has reduced the accuracy on old tasks.

5.2 Experimental setup

We train and test the model performance over metrics ACC and BWT with 6 different scenarios. First, we train the CNN architecture without adapting the learning rate for different tasks. Second, we train the architecture following the setting provided in UCB. Note that UCB do not report the results on CNN architecture for all three data sets we experiment with, hence, we tune the parameters and train the network with the approach provided in UCB. Finally, we train model on the proposed approaches namely channel-wise, group-channel-wise, filter-wise and layer-wise. For group-channel-wise approach, we divide the filter channels into 4 groups for every filter in every layer. We use train-validation-test approach for training, tuning and testing parameters and a common CNN architecture for all our experiments listed in Table 2. To test the structured adaptability bias hypothesis, we use Permuted MNIST data set to demonstrate that in the absence of any statistical significance in the parameters in spatial or depth dimensions of each layer, the imposed bias does not exhibit any improvements.

5.3 Implementation details

We used PyTorch and automatic differentiation for training. The initial learning α_0 for both μ and ρ was set to 0.01. μ was initialized using a standard normal distribution and ρ was initialized to -3.0 with the added Gaussian noise. The prior parameters σ_1 and σ_2 were set to 0.0 and 6.0 respectively. We used 10 samples of ϵ in each mini-batch training where mini-batch size was set to 64. Each experiment was run 5 times and averaged to report the results.

Table 2: Convolution architecture used for all the approaches and the UCB baseline with 5-Split MNIST and 5-Split Fashion MNIST. For the Permuted MNIST, the FC filter output shape is 10. s in Conv and Pool represents the stride and k in Pool represented the kernel size. Bias layer is not shown.

Operation type	Filter Shape	Input Size
Conv(s=1)+ReLU+MaxPool(s=2,k=2)	$1 \times 8 \times 3 \times 3$	$1 \times 28 \times 28$
Conv(s=1)+ReLU+MaxPool(s=2,k=2)	$8 \times 16 \times 3 \times 3$	$8 \times 14 \times 14$
Conv(s=1)+ReLU+MaxPool(s=2,k=2)	$16 \times 16 \times 3 \times 3$	$16 \times 7 \times 7$
Conv(s=1)	$32 \times 16 \times 3 \times 3$	$16 \times 3 \times 3$
Flatten + FC + Softmax (T heads)	288×2	288

6 Results

In this section, we present the results and draw conclusions from our experiments on the three different data sets. The performance result is shown in Fig. 2a, 2b and 2c.

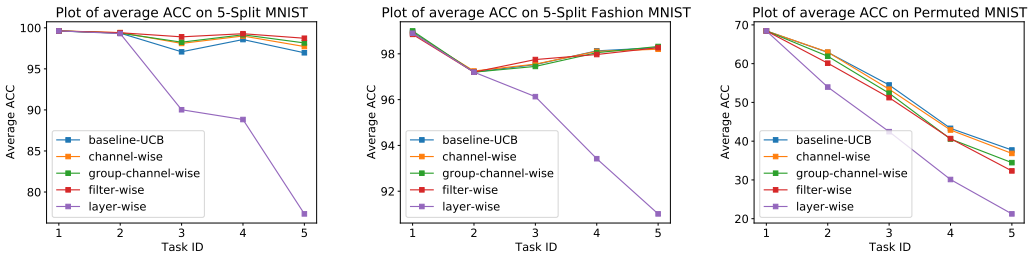
6.1 Storage complexity

Let L, F, C, M and N be the upper bounds for the number of layers, filters per-layer, channels per-filter, spatial size of a filter in horizontal dimension and spatial size of a filter in vertical dimension respectively. The parameter complexity for the baseline UCB is then given by $\mathcal{O}(LFCMN)$. We present the parameter complexity for our approach in table 3.

6.2 Results on split-5 MNIST, split-5 fashion MNIST and permuted MNIST

For MNIST and Fashion MNIST, we observe a slight improvement in model's average accuracy after training on 5 tasks for channel-wise, group-channel-wise and filter wise where filter-wise performance in statistically significant for MNIST data set. The layer-wise consistently performed poor which may be due excessive constraints put on uncertainty parameters. Moreover, the methods that performed well in terms of accuracy, they also possessed baseline equivalent backward transfer.

Permuted MNIST case is an anti-pattern because this data set is not meant for continual learning. Nevertheless, as shown in the figure 2c , the consistent degradation for all our approaches shows that if the block structure does not capture relevant features, then sharing uncertainty does harms the training. This works as an empirical validation for our *structured adaptability bias* hypothesis. Table 3 shows the complexity reduction in parameters and the performance metrics for all approaches.



(a) (b) (c)

Figure 2: Figure shows ACC on all approaches on three data sets. (a) ACC on MNIST, (b) ACC on Fashion MNIST, (c) ACC on Permuted MNIST.

Table 3: Uncertainty parameter storage complexity and model performance metrics (ACC and BWT) on three data sets (MNIST, Fashion MNIST and Permuted MNIST) with baseline and all our approaches is reported.

Approach	Parameter complexity	MNIST ACC	MNIST BWT	Fashion MNIST ACC	Fashion MNIST BWT	Permuted MNIST ACC	Permuted MNIST BWT
Without adaptation	$\mathcal{O}(1)$	60.2%	-39%	83.2%	-18%	27.2%	-48%
Baseline-UCB	$\mathcal{O}(LFCMN)$	96.96%	-2%	98.27%	0%	37.72%	-27%
channel-wise	$\mathcal{O}(LFC)$	97.72%	-2%	98.20%	0%	36.84%	-28%
group-channel wise	$\mathcal{O}(LFC_g)$	98.16%	-1%	98.32%	0%	34.46%	-30%
filter wise	$\mathcal{O}(LF)$	98.74%	0%	98.27%	0%	32.33%	-31%
layer wise	$\mathcal{O}(L)$	77.33%	-12%	91.01%	-4%	28.24%	-39%

7 Discussion and Conclusion

In this paper, we present a parameter efficient regularization-based continual learning approach that uses the Bayesian framework to learn parameter uncertainty in a neural network and use it as a proxy to adapt learning rate on a sequence of tasks. We exploit the convolution architecture to propose four different approaches that can improve storage complexity and model performance. Moreover, we formulated an experiment that demonstrated the correctness of our *structural adaptability bias* hypothesis. As a future work, this approach can be investigated for other architectures such as Recurrent Neural Network(RNN). The applicability of learning parameter uncertainty as proposed in this paper can be used in other novel use cases other than continual learning.

References

[1] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[2] Jathushan Rajasegaran, Munawar Hayat, Salman H Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for continual learning. In *Advances in Neural Information Processing Systems*, pages 12669–12679, 2019.

[3] Daniel L Silver, Qiang Yang, and Lianghao Li. Lifelong machine learning systems: Beyond learning algorithms. In *2013 AAAI spring symposium series*, 2013.

[4] Ralf Klinkenberg and Thorsten Joachims. Detecting concept drift with support vector machines. In *ICML*, pages 487–494, 2000.

[5] Davide Maltoni and Vincenzo Lomonaco. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116:56–73, 2019.

[6] Markus Wulfmeier, Alex Bewley, and Ingmar Posner. Incremental adversarial domain adaptation for continually changing environments. In *2018 IEEE International conference on robotics and automation (ICRA)*, pages 1–9. IEEE, 2018.

[7] Yu Hao, Yanwei Fu, Yu-Gang Jiang, and Qi Tian. An end-to-end architecture for class-incremental object detection with knowledge distillation. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2019.

[8] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *arXiv preprint arXiv:1909.08383*, 2019.

[9] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in neural information processing systems*, pages 6467–6476, 2017.

[10] Ju Xu and Zhanxing Zhu. Reinforced continual learning. In *Advances in Neural Information Processing Systems*, pages 899–908, 2018.

[11] Tianjun Xiao, Jiaxing Zhang, Kuiyuan Yang, Yuxin Peng, and Zheng Zhang. Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 177–186, 2014.

[12] Timothy J Draelos, Nadine E Miner, Christopher C Lamb, Jonathan A Cox, Craig M Vineyard, Kristofor D Carlson, William M Severa, Conrad D James, and James B Aimone. Neurogenesis deep learning: Extending deep networks to accommodate new classes. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 526–533. IEEE, 2017.

[13] Jose L Part and Oliver Lemon. Incremental on-line learning of object classes using a combination of self-organizing incremental neural networks and deep convolutional neural networks. In *Workshop on Bio-inspired Social Robot Learning in Home Scenarios (IROS), Daejeon, Korea*, 2016.

[14] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[15] David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. In Sheila McIlraith and Kilian Weinberger, editors, *The Thirty-Second AAAI Conference on Artificial Intelligence*, pages 3302–3309, United States of America, January 2018. Association for the Advancement of Artificial Intelligence (AAAI). AAAI Conference on Artificial Intelligence 2018, AAAI 2018 ; Conference date: 02-02-2018 Through 07-02-2018.

[16] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[17] Amal Rannen, Rahaf Aljundi, Matthew B Blaschko, and Tinne Tuytelaars. Encoder based lifelong learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1320–1328, 2017.

[18] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[19] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. *Proceedings of machine learning research*, 70:3987, 2017.

[20] Vicent Sanz Marco, Ben Taylor, Zheng Wang, and Yehia Elkhatib. Optimizing deep learning inference on embedded systems through adaptive model selection. *ACM Transactions on*

Embedded Computing Systems (TECS), 19(1):1–28, 2020.

[21] Hanna Tseran, Mohammad Emtiyaz Khan, Tatsuya Harada, and Thang D Bui. Natural variational continual learning. In *Continual Learning Workshop@ NeurIPS*, volume 2, 2018.

[22] Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. Uncertainty-based continual learning with adaptive regularization. In *Advances in Neural Information Processing Systems*, pages 4392–4402, 2019.

[23] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

[24] Ferenc Huszár. Note on the quadratic penalties in elastic weight consolidation. *Proceedings of the National Academy of Sciences*, page 201717042, 2018.

[25] Yijun Li, Richard Zhang, Jingwan Cynthia Lu, and Eli Shechtman. Few-shot image generation with elastic weight consolidation. *Advances in Neural Information Processing Systems*, 33, 2020.

[26] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

[27] Sayna Ebrahimi, Mohamed Elhoseiny, Trevor Darrell, and Marcus Rohrbach. Uncertainty-guided continual learning with bayesian neural networks. In *International Conference on Learning Representations*, 2019.

[28] Tameem Adel, Han Zhao, and Richard E Turner. Continual learning with adaptive weights (claw). In *International Conference on Learning Representations*, 2019.

[29] Menelaos Kanakis, David Bruggemann, Suman Saha, Stamatios Georgoulis, Anton Obukhov, and Luc Van Gool. Reparameterizing convolutions for incremental multi-task learning without task interference. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 689–707, Cham, 2020. Springer International Publishing.

[30] Pravendra Singh, Vinay Kumar Verma, Pratik Mazumder, Lawrence Carin, and Piyush Rai. Calibrating cnns for lifelong learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[31] Andrei A. Rusu, Matej Vecerík, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. In *1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings*, volume 78 of *Proceedings of Machine Learning Research*, pages 262–270. PMLR, 2017.

[32] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks.

[33] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.

[34] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[35] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018.

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

[37] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.