3rd Symposium on Advances in Approximate Bayesian Inference, 2020 1–7

# Conditional Input Gated Low-Rank Perturbations for Continual Learning

**Anonymous Authors**
*Anonymous Institution*

## Abstract

We address the problem of learning convolution neural networks (CNN) in the continual setting when tasks arrive sequentially, and only the data of the current task is available. In this setting CNNs are prone to reduce their quality on all old task drastically. In this work, we extend the idea of the (Abati et al., 2020) of data-conditional expansion of the CNN architecture. We propose to use low-rank and hence more weak additive perturbations for CNN filters, which is enough due to the composition structure of the CNN layers. Such low-rank adaptation modules allow us to reduce computational costs and promote sparseness in the adaptation of CNN to new tasks. We validate our approach empirically on the split MNIST and CIFAR10 tasks.

## 1. Introduction

Learning a discriminative model – classifier, from an empirical distribution, is one of the core tasks in machine learning. State-of-the-art methods for image classification imply the use of deep neural networks (DNN). The price of such success is training deep models on a large amount of data, which is assumed to reflect data distribution at the time of model application. In contrast, in real-world settings, we meet with *time-evolving distributions* of the data, such as new task appearing.

We are interested in methods that can adapt with datastream without a drop of performance on the previously learned tasks. Moreover, we usually can not afford to store all previous data and re-train model from scratch. However, DNNs are prone to drastically reduce their quality when trained in the continual setting on data from a new task, without access to the previous data. To this end, several approaches to overcome this phenomenon of catastrophic forgetting were suggested.

Weight regularization approaches add quadratic regularization to the optimization objective (Liu et al., 2018; Huszár, 2017; Kirkpatrick et al., 2017). Such penalty forms the trust-region around the optimum of the previous task while optimizing loss on new data. The model architecture remains the same. Hence, we can apply this approach with the implicit assumption on the similarity of new tasks. Another approach is based on the incremental update of the net architecture: increase in its capacity, while sharing feature extractors across tasks (Rusu et al., 2016; Li and Hoiem, 2018). As a disadvantage of such methods, we consider unnecessary expansion of the net architecture if tasks share a lot in common.

We suppose that amount of network expansion should be learnable directly from the data. Recently (Abati et al., 2020) proposed to use the task-specific gating mechanism for conditional computation. Each layer in the feature extractor is equipped with the small

additional NN, which takes activation and output binary mask. Hence, this NN selects filters sufficient for current task in a data-driven manner. Gate execution patterns allow for identifying critical filters for each task, which can be frozen to sustain the ability to solve preceding tasks.

To efficiently use the model's capacity, such critical sets should contain as few filters as possible. To decrease their sizes, (Abati et al., 2020) introduced the sparsity objective. However, since this sparsification procedure requires a prolonged number of epochs (400 epochs per task for Split MNIST using 3-layered CNN and 200 epochs per task for Split CIFAR-10 using modified ResNet-18) and must be done for each task individually it consequentially results in longer training time.

To extend this idea and mitigate drawbacks, we consider a different parametrization of the conditional computation. Based on the composition structure of the DNNs, we are arguing that even a small *pertrubation* of the layer output could be sufficient to adapt the whole network for new tasks. As such perturbation, we add low-rank separable convolution to the basic convolutional layer. The gating module controls the number of applied additive adaptation kernels to the basic convolutional layer. Such parametrization allows us to re-use learned information in a more computationally efficient way.

Our main contributions are the following:

- We investigate low-rank parametrization of small pertrubations of the convolution filter in order to adapt DNN for new task.
- We empirically validate the proposed algorithm on the disjoint image classification task at the Split MNIST and Split-CIFAR-10 datasets. Our parametrization leads to much more sparse adaptation, compared to the (Abati et al., 2020), while keeping the performance of the approach in same level.

## 2. Methods

We consider a sequence of disjoint classification tasks $T_i,\ i \in \{0, \dots, N\}$. We want to solve them by using a DNN, which is the composition of $j$ learnable convolutional layers $F^j$ and non-trainable non-linearities between them. The straightforward approach is to learn $n$ different models for each task: $\{F_i^j\}_{i=0}^N$, which is to ignore any similarities between tasks. Therefore, such an approach is neither computational nor data efficient. As opposite, we could train only one neural network $F_0^j$ on the first task data and apply it to all other tasks as well. As $F_0$ was optimized to extract features for the data in the task $T_0$ only, it may and will give bad representations for the data associated with $T_i$ task. Again, this approach is not to utilize common information across the tasks.

However, these approaches and composition structure of the DNN motivate as to adapt $F_0$ to $T_i$ by freezing $F_0$ them selves and introducing task-specific learnable adaptation modules $F_i^p$ in such form, that $F_i = F_0 + F_i^p$. If $F_i^p$ is small kernels and applied *rarely*, it would be both data and computationally efficient solution. To this end, we propose to use gate-mechanism which decides, to which object and channel to apply $F_i^p$. Next, we propose to learn only low-rank separable convolutional kernels $F_i^p$. This combination with sparsity objective over the number of used $F_i^p$ leads to an efficient solution.

Next, we describe in details this main propositions: how to parameterize $F_i^p$ and if there is a strategy to apply them in a data-driven manner.

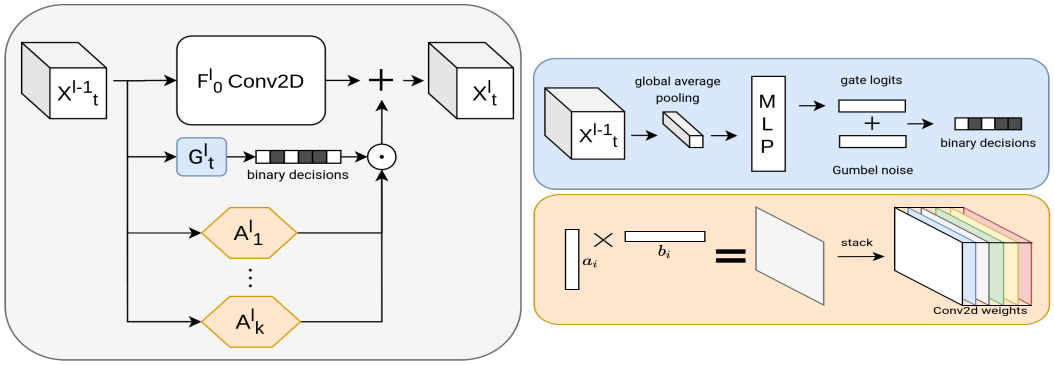



Figure 1: Scheme of the modified layer $l$ of the base model $F_0$. $A_i$ denotes individual adaptation modules, $k$ modules a0re deployed for the illustrated layer. $F_0^l$ denotes the conv. layer of the model $F_0, G_t^l$ denotes gate module for particular layer $l$ for particular task $t$.

**Parameterization of the Perturbation $F_i^p$** For each layer $j$ in the backbone $F_0^j$ we have several (we considered 1, 2, 4 and 8) number of adaptation modules. We want them to be efficiently computed and light. After experiments we ended up with two types of parameterization:

- Low rank matmul – Weight tensor is obtained via stacked low-rank matricies. The scheme of forming such modules is depicted on Fig. 1.
- ReLU separated spatial separable convolutions – This type uses classical spatial separated convolution, but in between of two convolutions ReLU is placed.

**Gate mechanism** We equip each convolutional layer in the feature extractor with a gate mechanism to employ adaptation modules in a data-driven manner. It is a shallow MLP followed by a *Gumbel-Softmax sampling* (Jang et al., 2016) to get binary masks still preserving differentiability of the computation graph. The scheme of forming such modules is depicted on Fig. 1. This mask indicates, which adaptation modules will be applied to the particular object in the batch. Then appropriate adapted representations (which are outputs of the particular adaptation modules) are selected and added to the output of the frozen layer of the base model. We consider different levels of fidelity for the mask:

- Object-wise – mask indicates, for which objects the sum of all adaptation modules will be applied. Resulting mask has the shape of [batch size $\times$ 1]
- Channel-wise – for each object mask indicates channels in the output tensor, to which the sum of all adaptation modules will be applied. Resulting mask has the shape of [batch size $\times$ # of channels in the tensor]
- Perturbation-wise – for each adaptation module the mask indicates, to which channels of the output tensor this module will be applied. Resulting mask has the shape of [batch size $\times$ # of channels in the tensor $\times$ # of adaptation modules]

| | MNIST | | | | | CIFAR10 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Model | $T_1$ | $T_2$ | $T_3$ | $T_4$ | avg | $T_1$ | $T_2$ | $T_3$ | $T_4$ | avg |
| *Base model refit (UB)* | 0.998 | 0.999 | 1.00 | 0.996 | 0.998 | 0.936 | 0.965 | 0.985 | 0.983 | 0.967 |
| EWC-On | 0.994 | 0.934 | 0.982 | 0.932 | 0.961 | 0.804 | 0.803 | 0.952 | 0.960 | 0.879 |
| LwF | 0.979 | 0.997 | 0.999 | 0.985 | 0.990 | 0.873 | 0.671 | 0.505 | 0.514 | 0.641 |
| HAT | **0.996** | 0.999 | 0.998 | 0.990 | 0.996 | 0.911 | **0.953** | **0.985** | 0.977 | 0.957 |
| CCGN | 0.994 | **1.00** | 0.999 | **0.993** | **0.997** | 0.917 | 0.950 | 0.983 | 0.978 | 0.957 |
| **Ours** | **0.996** | **1.00** | **1.00** | 0.992 | **0.997** | **0.936** | 0.941 | 0.982 | **0.984** | **0.961** |

Table 1: Task-incremental results. Our approach out performs competitors with static architecture and shows better or comparable results with other dynamic-architecture approaches, while using less parameters.

## 3. Experiments and Discussion

We have a sequence of n classification tasks $T_i,\ i \in \{0, \ldots, n-1\}$. For task $T_0$ we learn model $F_0$. Then for each upcoming task We perform the following modification of the $F_0$: we freeze each convolutional layer $l$ in $F_0$ and add $k$ task-specific perturbations $A_i^l$ and gate module $G^l$. Thus we will be training gates and adaptation modules only.

**Datasets** We evaluated our approach on Split MNIST(LeCun and Cortes, 2010) and Split CIFAR-10(Krizhevsky, 2012) datasets. Each dataset was split into 5 subsets of consecutive classes. This results into 5 binary classification tasks that are observed sequentially. The reported metric is accuracy on the hold-out test set.

**Base models** As a base model $F_0$ for Split MNIST benchmark we considered A simple Convolutional Neural Network consisting of three layers with 64, 64, 128 output channels respectively. Pooling layers were deployed in the following order: (MaxPool $2 \times 2$, AvgPool $2 \times 2$, GlobalAvgPool). Gating is applied before the pooling layer. A final linear classifier yields class predictions. For Split CIFAR-10 we used ResNet-18, modified according to the procedure, described above.

**Concurrent approaches**

- Base model refit – each task was learned with individual new instance of the base model $F_0$. We considered performance of this scheme as the upper bound.
- EWC-On(Schwarz et al., 2018) – the online version of Elastic Weight Consolidation.
- LwF(Li and Hoiem, 2018) – regularization of the task loss by a distillation objective.
- HAT(Serrà et al., 2018) – mask-based model conditioning the active units in the network on the task label.
- CCGN(Abati et al., 2020) – conditional channel gated network which operates the the capacity via gate module.

Results for all concurrent approaches except base model refit were taken from the (Abati et al., 2020).

### 3.1. Experiments and Results

For Split MNIST benchmark we used 1 adaptation unit per layer and object-wise gate mode. Each perturbation was calculated according to the Low rank matmul strategy, described in the Section 2. For Split CIFAR-10 we employed 8 adaptation units per layer and perturbation-wise gate mode, which ensured the most effective use of each unit. Here perturbations were calculated as ReLU separated spatial separable convolutions. We also upsample all images to the resolution of $112 \times 112$. The joint results of both experiments are presented in the table 1. We observe in different datasets that our solution achieves better or comparable results with other methods mentioned. The main idea of our solution is data drive architecture expansion. Hence our main competitor is (Abati et al., 2020). In comparison with this work, we use fewer parameters and fewer perturbations.

### 3.2. Gate ablation study

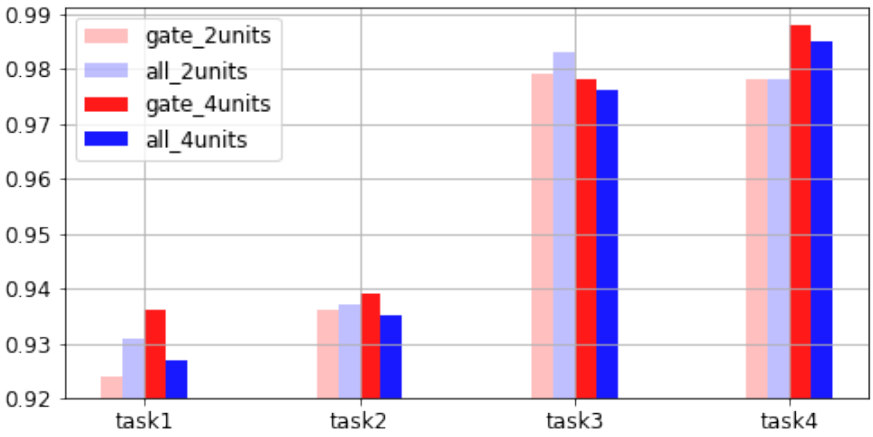



Figure 2: Ablation study.

### 3.3. Different gate modes

To justify the use of the gate mechanisms, we compared two models on the Split CIFAR10 dataset. The first one (on Fig. 2) was forced to apply all adaptation modules for each object in the batch, whereas the second one (red) took advantage of the gate mechanism. Results are presented on Fig. 2. For a lower number of adaptation units, the gate mechanism proves to be excessive. However, as we expand the number of variants of adaptation module to apply selection across them leads to better results. Note, that *ability* to select from four perturbations does not mean that gate mechanism select to apply all four to each object. In particular, we observe application of low number of different perturbations to different objects.

We tested different gatemodes, described above in Par. 2, on the Split CIFAR-10 dataset. Results are presented on Fig. 3.3

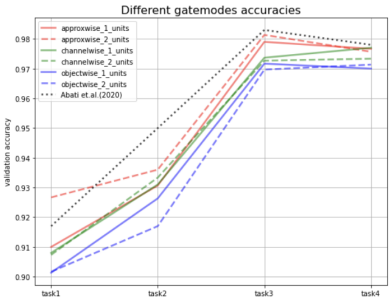



Figure 3: Different gatemodes' performances. Modes illustrated by color, # of units - by line style. The results of (Abati et al., 2020) were added for a better visual comparison.

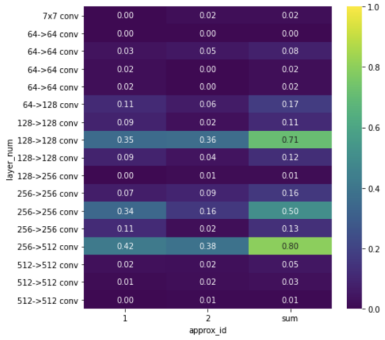
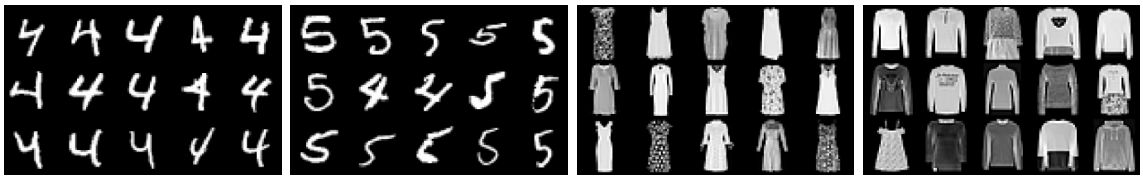



Figure 4: Adaptation modules' execution pattern with strong sparsity objective regularization. For each module (represented by the *approx_id*) in each layer (represented by the *layer_num*) frequency of being applied is reported.

**Gate sparsities** We introduce the sparsity objective as in (Abati et al., 2020) to encourage gate to choose the minimal set of approximations needed for solving the task. Our study shows, that without any significant loss of accuracy the adaptaton modules' usage can be quite sparse. As depicted on Fig. 4, the net leaves early layers untouched and selects several layers in the middle, where perturbations are applied extensively.



$(a)$ $(b)$ $(c)$ $(d)$

Figure 5: (a) and (b) represents MNIST examples of gates 0/1 states, (c) and (d) FashionMNIST

**Gate generalization in objectwise mode** In objectwise mode for each image gate decides, if the perturbations will be added or not. This sometimes leads to the interesting patterns in gate's decisions For example, gate sometimes to distinguish classes and apply perturbations to the particular one as depicted on Fig. 5

## 4. Conclusion

We propose the approach of mitigating catastrophic forgetting by the data-driven expansion of the network architecture. For each object to classify the gate-mechanism makes the binary decision of adaption module application. The binary decision relaxed with Gumbel-Softmax sampling. To re-use common information between the sequence of the task, we parametrize adaptation modules with low-rank perturbations of Conv. filters.

## References

Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. pages 3930–3939, 06 2020. doi: 10.1109/CVPR42600.2020.00399.

Ferenc Huszár. On quadratic penalties in elastic weight consolidation. *arXiv preprint arXiv:1712.03847*, 2017.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

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.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018. doi: 10.1109/TPAMI.2017.2773081.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2018.

Xialei Liu, Marc Masana, Luis Herranz, Joost Van de Weijer, Antonio M Lopez, and Andrew D Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2262–2268. IEEE, 2018.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Jonathan Schwarz, Jelena Luketina, Wojciech M. Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress compress: A scalable framework for continual learning, 2018.

Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task, 2018.