Optimizing Reusable Knowledge for Continual Learning via Metalearning [1]

Pranjal Aggarwal, Prerna Agarwal

2019CS50443, 2021AIZ8328

1 Introduction and Motivation

In Deep Neural Networks, memory can be lost when new knowledge disrupts old information during task inference and this problem is known as Catastrophic Forgetting (CF). This problem is more evident in continual learning due to the unavailability of previous data in sequential set of tasks. Previous work to prevent CF either avoids modification of parameters that are key to solve previous tasks or includes methods that reserve part of the network to learn new task and use special memory units to recall critical training examples from previous tasks. The main problem with these methods is the extra model complexity and requirement of an efficient method to recall key information from previous tasks.

This paper [1] takes inspiration from human learning mechanism while learning new tasks. Humans tends to associate previous experience to new situations and strengthen previous memories that helps to overcome CF problem. The authors propose a new model based on a learning strategy that, instead of mitigating weight overwriting or learning independent weights for different tasks, uses a metalearning approach for weight reusability among tasks using a shared KB. KB is not given by an external memory that encodes information in its vectors but by a trainable model that encodes shared information in its weights.

2 Summary of the Work

The summary of this work is as follows: This work tries to overcome the problem of catastrophic forgetting due to overwriting of weights during training of a new task (in a Continual Learning scenario). For this, the authors have propose a new method - Meta Reusable Knowledge that uses a meta learning approach for weight reusability by keeping a set of shared weights among tasks instead of overwriting while learning a new task. These shared weights becomes a common knowledge base that is incrementally enriched with new knowledge as the model learns new tasks. This KB is a trainable model that encodes shared information in its weights. To query this KB, a set of trainable masks are present that allows selectively choose relevant weights from KB to solve a task. The approach force the model to reuse current knowledge in KB by detecting importance of each pattern learned in KB and expand it's knowledge if the current knowledge is not sufficient to learn a task. Their experiments shows that the use of task dependent masks are critical in achieving good performance. The authors use episodic training meta learning strategy to train KB from data. On top of KB, mask generating functions are trained for each task. These masks are queries that helps in accessing the intermediate activations of the KB. These masks are depended only on each specific task and input. These masks are fully connected layers. Given an input, the corresponding mask is used to query the KB to generate a feature vector which can then be used by a task dependent classification head to provide prediction.

3 Implementation and Experimentation Details

3.1 Implementation Details

We have implemented this paper using python PyTorch framework. The codebase with Jupyter notebook and run instructions is available here: `https://github.com/Pranjal2041/MARK-Extended`. We first created a MARK-model structure consisting of a list of Feature Extractors for each task, KB, Masks and Classifiers for each task. These modules are implemented in the following steps :

1. *Feature Extractors*: The feature extractors provide an initial embedding for the input. For each task, feature extractor F_t consists of 2 convolutional layers with batch normalization and ReLU with max pooling as specified in the paper for image task and LSTM for market CL data.
2. *Initialize KB*: The KB is shared across tasks and accumulates knowledge with every new task. It also consists of three convolutional layers as specified in the paper for image task and similarly 3 layer LSTM for market CL data.
3. *Train Mask Generator and Classifier*: Feature vector extracted by feature extractors are used to create mask M_t for each task. They select what knowledge is relevant for each task. It is implemented using fully connected layers as specified in the paper. The task dependent classifier C_t takes input from mask conditioned KB and produce prediction.
4. *Update KB*: A metalearning approach is used to update the KB where the inner loop updates the KB on mini-batches of the data. The outer loops gets updated with the aggregation of the updations in the inner loop.
5. *Fine-Tune Mask Generator and Final Classifier*: After KB is updated at every task, the mask M_t and classifier C_t is fine-tuned.

3.2 Experimentation Details

We evaluate the approach using two metrics specified in the paper: average accuracy (Acc) and backward transfer (BWT). We also use an additional metric F1-Score to evaluate the approach. Acc will measure the average performance after learning on T tasks. BWT would measure how much performance is lost on previous tasks after learning on new tasks.

We experiment with the following datasets: CIFAR-100 (used in the paper), Image Data (not used in the paper) and Market CL (not used in the paper).

For CIFAR-100, we experiment with the specifications mentioned in the paper: Stochastic Gradient Descent as optimizer with a learning rate of 0.1 for steps 1-4 and 0.05 for step 5. Batch size of 128 is used. A total of 20 tasks are trained for 50 epochs each in step 1, 2 and 3. For step 4, it is trained for 30 epochs. The hidden layer dimensions of convolutional layers are [64, 128, 256]. To update the KB, we use 10 mini batches/meta-tasks (K), trained for 40 epochs for the inner loop and 15 epochs for outer loop.

For ImageData also, the same specifications were used except the number of tasks being 5 here.

For MarketCL, the problem is formulated as follows:

- For each stock symbol and a day, 375 mins of stock trading data has been provided. To create the data points, we encode this sequential information with multi-layer LSTM network as feature extractors.
- The model for all symbols are updated for each day using the datapoints created at the above step.
- The fully connected layer of classifiers contains 4 heads - one for each label prediction.

For Market CL, we experiment with the following specifications: Stochastic Gradient Descent as optimizer with a learning rate of 0.001 for steps 1-4 and 0.005 for step 5. Batch size of 4 is used. A total of 20 tasks (sampled randomly) are trained for 16 days and for 30 epochs each in step 1, 2 and 3. For step 4, it is trained for 25 epochs. The hidden layer dimensions of LSTM layers are [64, 128, 256].

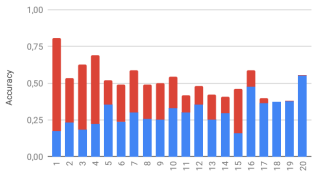
4 Baselines

The following baselines have been used. First 3 of them are obtained from the paper. The last one have been used only for MarketCL dataset, since it made sense for that dataset only.

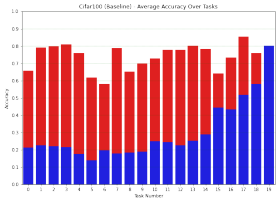
1. **Baseline1 (Baseline):** Simple sequential learning with no metalearning or mask-generating functions. Architecture used was same as that of KB.
2. **Baseline 2:** Baseline + Meta Learning in KB Update.
3. **Baseline 3:** Baseline + task-specific Mask Generating Function.
4. **Baseline 4:** 20 different models were trained i.e a different model each symbol separately. This baseline was implemented only for MarketCL dataset.

4.1 Baselines for CIFAR-100

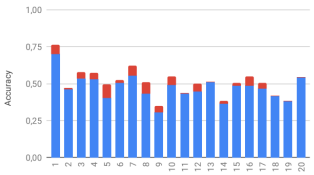
The Figure 1 shows the baselines results from our implementation and compared with the baseline results reported in the paper.



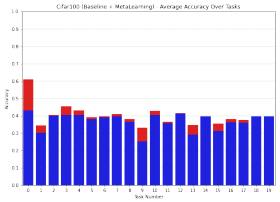
(a) Baseline from Original Paper



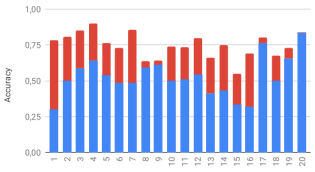
(b) Baseline Recreated



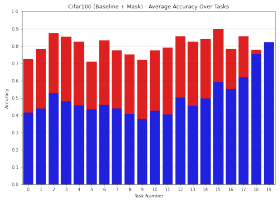
(c) Baseline + MetaLearning Original Paper.



(d) Baseline + MetaLearning Recreated



(e) Baseline + Mask Original Paper.



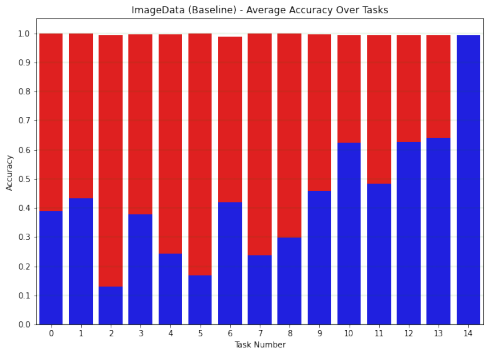
(f) Baseline + Mask Recreated

Figure 1: Baselines for CIFAR-100

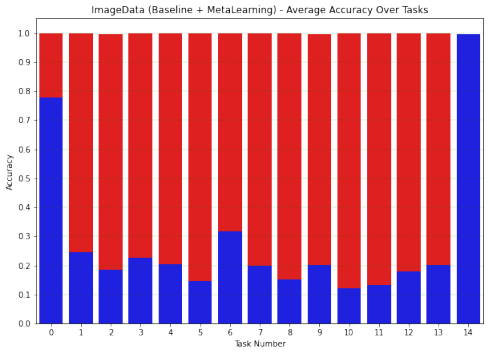
Since, hyperparameters for baselines weren't available, it was difficult to recreate the same numbers, however, they still come out to be very close to the original paper. Specifically the trends mentioned in the paper are similar to those in our recreation, thus, verifying the claims in paper.

4.2 Baselines for ImageData:

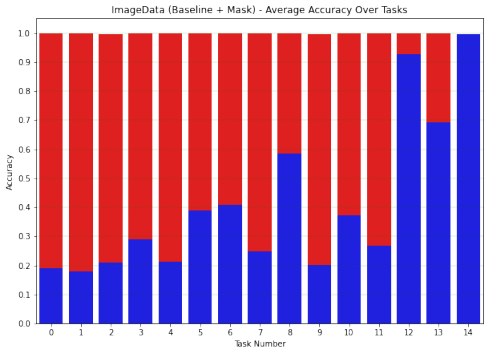
This dataset is not used in the paper. As shown in Figure 2, for all 3 baselines we observe that the forgetting is very high. Also, the task in hand is very simple, therefore, for each new task learned - initially it had almost 100% accuracy, though it decreases as forgetting happens.



(a) Baseline Recreated



(b) Baseline + MetaLearning

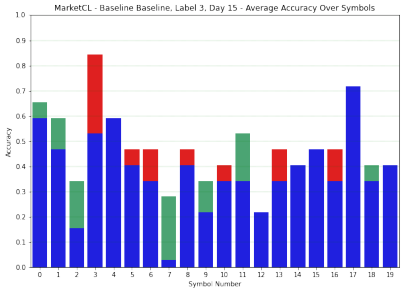


(c) Baseline + Mask

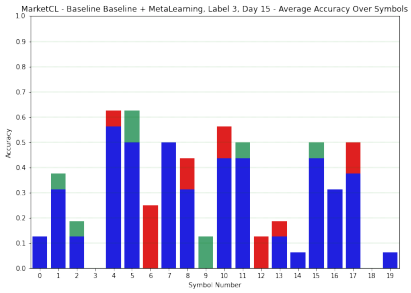
Figure 2: Baselines for ImageData

4.3 Baseline for Market CL Data:

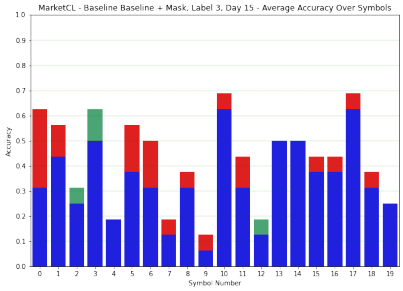
This dataset is also not used in the paper. For simplicity, Figure 3 only shows baselines for day 15, and label 3[(0.005, 0.01)].



(a) Baseline Recreated



(b) Baseline + MetaLearning



(c) Baseline + Mask

Figure 3: Baselines for MarketCL

For Baseline 4, Accuracy achieved was 42.812% with BWT of 0 (since different model was used for each symbol). Overall, Basline 4 performs the best (at cost of larger model size) and from now on, only this baseline will be compared for MarketCL.

5 Results

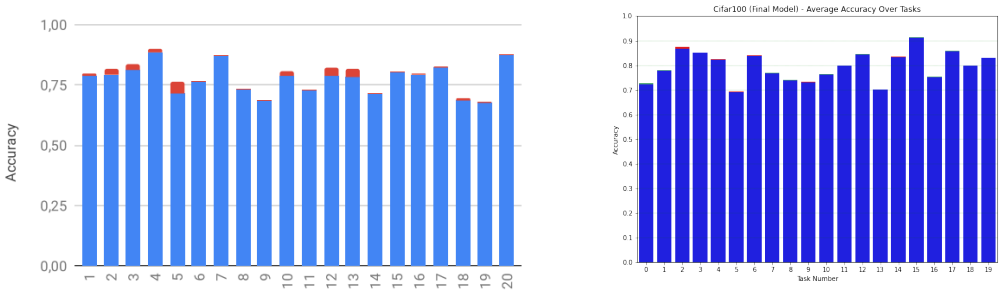
This section shows the the results obtained using the full MARK model proposed in the paper for different datasets.

5.1 CIFAR-100

5.1.1 Observations and Discussion

- We observe that the proposed MARK model approach clearly outperform the baselines, in terms of both backward transfer and accuracy.
- Secondly, we see that our implementation results are in better numbers than that of reported in paper. However, we have followed the same implementation as in paper, and have adhered to hyperparameters wherever reported in paper. Thus, the small change in numbers could have been due to implicit changes in parameters and structure of models.
- Another important thing observed during course of experiments was: For the given configuration meta-learning wasn't really important. Basically, as an ablation study, we trained KB only for the first task using traditional training. On new tasks, KB wasn't trained any further. Yet we got very comparable numbers i.e 80.13% accuracy with 0 BWT. To verify our claims, we also tested this on author's open source code directly, and same observation was seen. *The main conclusion from this is that, for the datasets used in paper, meta-learning may not be of much avail.* There are two reasons for this:

  1. The training on first task of CIFAR-100, was sufficient to teach the knowledge base sufficient information about image features in general.
  2. The mask generating functions further helped in selecting the right weights from the knowledge base. This is because, using suitable masks on even randomly initialized weights(here the case was even better) can show good accuracy.
  3. since it was task specific there was no chance of forgetting. While this was true for CIFAR-100, this cannot be true for Market-CL, because it is a very different task, and rather the knowledge base helps in learning shared information, which helps in even positive backward transfers.



(a) MARK Model from Paper (b) Our Implementation

Figure 4: MARK Model Results on CIFAR-100

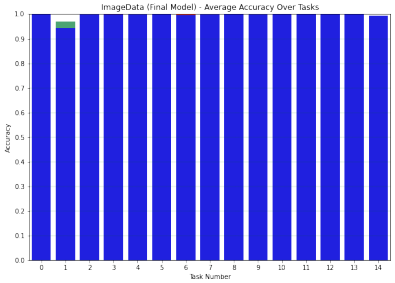
Method	**CIFAR-100**	
Metrics	Acc (std.)%	BWT%
Baseline 1	$30.61(\pm 0.904)$	-34.76
Baseline 2	$39.62(\pm 0.511)$	-5.19
Baseline 3	$30.69(\pm 1.008)$	-18.47
MARK-Paper	$78.31(\pm 0.3)$	-0.27
MARK-Our implementation	$\mathbf{80.36}(\pm 0.313)$	$\mathbf{-0.025}$

Table 1: Results on CIFAR-100

5.2 ImageData

5.2.1 Observations and Discussion

- The MARK model clearly outperforms the baseline models.
- The accuracy of mark model is very high. The reason is that the dataset was fairly simple, with only 400 pixels per image. On top of that, the 74 original classes, were divided into 15 tasks with 5 classes each. Thus, the task specific modules had to learn to only distinguish between 5 classes which is fairly simple. Thus such high accuracy was observed.



(a) Our Implementation

Figure 5: MARK Model Results on ImageData

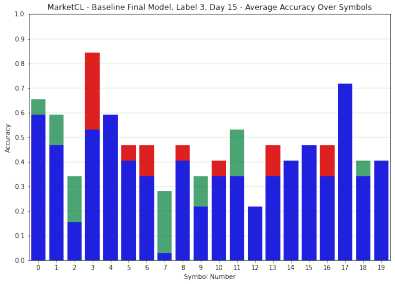
Method	**ImageData**	
Metrics	Acc (std.)%	BWT%
Baseline 1	$40.11(\pm 3.92)$	-53.55
Baseline 2	$23.16(\pm 1.41)$	-73.53
Baseline 3	$42.38(\pm 2.94)$	-61.57
MARK-Our implementation	$\mathbf{99.77}(\pm 0.06)$	$\mathbf{-0.0022}$

Table 2: Results on ImageData

5.3 MarketCL

5.3.1 Discussion

- We observe that BWT is positive for many of the symbols. This indicates knowledge culmination in KB is indeed helping and multiple tasks are helping each other in learning. Average BWT over all labels and sub-classes is +0.657%.



(a) Our Implementation

Figure 6: MARK Model on Market CL- Label 3

Method	MarketCL			
Metrics	Label 1 Acc. %	Label 2 Acc. %	Label 3 Acc. %	Label 4 Acc. %
Baseline 4	43.35	50.33	**40.15**	**40.04**
MARK- Ours	**45.70**	**53.63**	38.60	39.04

Table 3: Results on Market CL

- Despite the fact that baseline 4 is using different models for different tasks, we see that overall accuracy is comparable to the MARK model. Infact, in 2 of the 4 labels, MARK has performed better than this baseline. The reason for this is partly because of cross task learning.
- Mark Model is much better than this baseline in terms of model size, because the task specific modules are relatively smaller in size. Moreover, in real world setting, there would be even more symbols, where the advantage of MARK model could be seen more evidently.

6 Limitation and Improvement over current Approach

The current approach is heavily dependent on task specific feature extractors, masks and classifiers. Therefore, the parameters scales linearly with the number of tasks, making it infeasible to use it for practical applications. This is observed even in the case of Market CL data which had 308 symbols and it was getting difficult to train the model.

In order to overcome this limitation to some extent, we propose to use the concept proposed in Supermasks in Superposition (SupSup) paper [2] to train only one classifier. We experiment the SupSup approach only to combine the task specific classifiers and mask generator to one classifier and mask generator only and not with feature extractors. This is because, a pre-trained feature extractor (e.g., ResNet for image data) can always be used to get rid of task specific feature extractors. We believe that with this improvement in scalability of the model, the performance degradation is negligible (demonstrated) while reducing the increase in the parameters w.r.t the number of tasks to a great extend. The results are formulated in the following table:

Method	MarketCL			
Metrics	Label 1 Acc. %	Label 2 Acc. %	Label 3 Acc. %	Label 4 Acc. %
MARK - Ours	**45.70**	53.63	**38.60**	**39.04**
MARK + SupSup	43.72	**56.33**	24.19	31.60

We observe here that the accuracy for first 2 labels is almost comparable. However, for Label 3 and Label 4 accuracy is significantly lower. We attribute this to lack of hyperparameter tuning. This is because the experiments on SupSup were performed at the last, and due to lack of availability of enough compute hardware, the required training to reach best numbers wasn't possible. Nonetheless, the results demonstrate that using SupSup along with MARK can have promising results specially when scaling to 1000s of symbols. Further with sufficient hyperparameter tuning, accuracies comparable to MARK model can be achieved, with much lesser number of model parameters.

7 Conclusion and Future Work

This paper propose a method that learns a meta learned KB to incrementally accumulate knowledge from different tasks in continual learning scenario to mitigate Catastrophic Forgetting. The method also use mask-functions to query the KB and provides a boost in the performance. We experimented this approach with one of the dataset used in the paper and 2 other datasets i.e., ImageData and MarketCL. For the former one, we showed similar results to original paper, and also identified a potential missing study in the original paper. For Market CL dataset, we used LSTM to encode the sequential information of every minute. We showed that numbers were very good, and comparable to

using different models for each task. Infact we showed that for some of the symbols, the numbers were even better than using different models, thus indicating different symbols are helping each other to learn. Next we identified the issue of scalability of MARK model with number of tasks. In order to reduce this effect, we incorportated SupSup Model in our implementation, and showed that even with limited space and common module for whole model, good numbers can be achieved.

As future work, we plan to experiment with Transformers as well to see if it can further improve the performance. We also plan to experiment with pre-trained feature extractors such as ResNet to eliminate feature extractors for each task and thus, decreasing the model parameters further. Moreover, due to hardware limitations, many things werent experimented exhaustively. For example, training was done on only 20 tasks out of 308. Moreover, enough hyperparameter tuning on SupSup Extension couldn't be done. In future we aim to perform these tasks, and to study the effects of these variations, and show effectiveness of our proposed approach(s).

References

[1] Julio Hurtado, Alain Raymond-Saez, and Alvaro Soto. "Optimizing Reusable Knowledge for Continual Learning via Metalearning". In: *CoRR* abs/2106.05390 (2021). arXiv: 2106.05390. URL: https://arxiv.org/abs/2106.05390.

[2] Mitchell Wortsman et al. "Supermasks in Superposition". In: *CoRR* abs/2006.14769 (2020). arXiv: 2006.14769. URL: https://arxiv.org/abs/2006.14769.