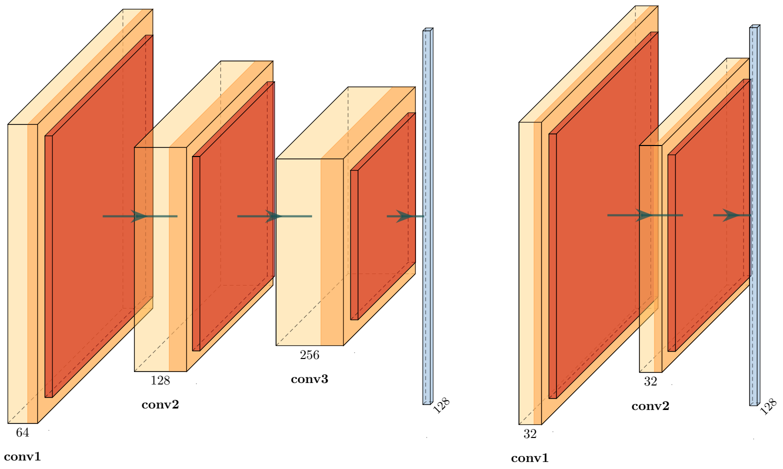
A Appendix

A.1 Architecture

In this section, we review in detail the architectural choices we made for this work. As we mentioned previously, we impose no restriction on which models we can use on each part of MARK. Please, see Figure 6 for an schematic diagram of the architectures we used. In our case, for each component:

- **Feature Extractor** (F^t): As we explained in the main document, this feature extractor can be any model, pretrained or otherwise. In the case of MARK-Task and MARK-Random, we use an architecture similar to the one used in the private model in [18], consisting of 2 convolutional layers with an activation function, batch normalization, and Max-Pooling, followed by a fully connected layer with ReLU. The number of filters in each layer depends on the dataset used. For CIFAR-100 we use 32 filters per block, while for MiniImageNet we use 8 filters per block. These numbers are similar as those used by ACL.
- **Knowledge Base (KB):** In this case, we use 3 shared blocks, each composed of 1 convolutional layer with 64, 128, and 256 components respectively, accompanied by an activation function (ReLU) and Max-Pooling. After these blocks, we add a fully connected layer that creates a representation to be input into the task classifier.
- **Mask-Generating functions (M^t):** As explained in the main document, we use a fully connected layer for each function accompanied by an activation function (ReLU). Input dimensions depend on the dataset, output is $(64 + 128 + 256) = 448$.
- **Classifier (C^t):** Is a fully connected layer.



(a) Knowledge Base

(b) Feature Extractor, F^t

Figure 6: Schematic representation of the architectures used in MARK.

A.2 Experiment Details

We run all of our experiments using 3 different seeds and implemented using PyTorch. In terms of hyperparameters, we use SGD with a learning rate of 0.01, weight decay of 0.01, a momentum of 0.9 and a batch size of 128. We set the momentum to 0 when updating the KB.

The F^t are trained for 50 epochs each when using MARK-Task. During the KB querying phase, mask functions and classifiers are trained for 50 epochs. In the KB Update step, we use 10 meta-tasks (K), trained for E_{inner} =40 epochs, each with a learning rate of 0.001. We repeat this training stage 15 times (E_{outer}). Our code will be made publicly available.

We also list the entire set of hyperparameters used for all methods which are compared against MARK. We use no learning rate scheduling for any of these experiments.

Table 2: Hyperparameters used for all methods compared against MARK

Method	Dataset	Learning Rate	Epochs	Optimizer	Batch Size	Result Origin
HAT	CIFAR100	0.01	As ACL	SGD	128	ACL Paper
A-GEM	CIFAR100	0.01	100	SGD	128	Avalanche[52]
ACL	CIFAR100	0.01	200	SGD	128	ACL Paper
GPM	CIFAR100	0.01	100	SGD	128	Paper Code
Exp. Replay	CIFAR100	0.01	100	SGD	128	Avalanche[52]
SupSup	CIFAR100	0.001	100	SGD	128	Paper Code
Multitask	CIFAR100	0.01	100	SGD	128	Code
HAT	MiniImageNet	0.01	As ACL	SGD	128	ACL Paper
A-GEM	MiniImageNet	0.01	100	SGD	128	ACL Paper
ACL	MiniImageNet	0.01	200	SGD	128	ACL Paper
GPM	MiniImageNet	-	-	-	-	GPM Paper
Multitask	MiniImageNet	0.01	100	SGD	128	Paper Code

A.3 Some notes for particular methods

- **A-GEM:** though A-GEM is originally run as a single pass algorithm, both on the ACL paper and on our own experiments ran it for multiple epochs to make results comparable. On our implementation for CIFAR100 we used a memory size of 200 per task.
- **Experience Replay:** We used a memory of 250 elements per task. This number is taken to make sure memory requirements between this baseline and MARK are similar.
- **SupSup:** We tried different values of sparsity until we were able to replicate the same results of their paper with their setup. Using that sparsity value, we replaced their model with a model similar to our KB but with extra parameters, similar to the model used in ACL and GPM. We tried to use 0.01 as learning rate, but results were much worse.
- **GPM:** For MiniImageNet, we were unable to find hyperparameters used either on the paper or code base.

A.4 CO2 Emission Related to Experiments

Experiments were conducted using a private infrastructure, which has a carbon efficiency of 0.432 kgCO_2eq/kWh. A cumulative of 75.5 hours of computation was performed on hardware of type GTX 1080 Ti (TDP of 250W).

Total emissions are estimated to be 8.16 kgCO_2eq of which 0% were directly offset.

We run 3 experiments on a single GPU at the same time. Experiments on CIFAR-100 take 133 minutes, while on MiniImageNet take 622 minutes. We consider 6 variations for both datasets, each run with 3 different seeds.

Estimations were conducted using the MachineLearning Impact calculator presented in [53].

A.5 Parameters

Table 3 reports number of parameters for models used by different methods. Numbers for HAT, A-GEM and ACL are taken from the respective articles. GPM numbers were calculated using the author's code.

A.6 Importance of Metalearning and Mask Functions

We supply an augmented version of Figure 5 in Figure 7 which includes data from MARK-Random and MARK-Resnet. The figure shows that using different input representations for the mask-generating functions has direct impact on the method's accuracy, but has little to no impact on BWT.

We hypothesize this behaviour is taking place because MARK's masks are acting as a kind of self-attention mechanism. With richer features, this mechanism is able to select more appropriate information to solve any given task. However, as expected, richer features aid nothing in attenuating

Table 3: Model Parameters for different methods in the Catastrophic Forgetting literature. HAT, A-GEM, ACL numbers were taken directly from the respective articles. GPM, SupSup were calculated based on author's code. Experience Replay was calculated based on our own code.

Method	CIFAR100	Mini-Imagenet
HAT	6.8M	30.9M
A-GEM	6.4M	25.7M
ACL	6.3M	28.3M
GPM	2.4M	1.5M
SupSup	20.9M	-
Experience Replay	2.9M	-
Multitask	7.4M	15.4M
MARK-Random	1.7M	6.7M
MARK-Task	4.7M	15.6M
MARK-Resnet	16.1M	19.7M

Catastrophic Forgetting. The role of features is to provide information with which to decide, while Backward Transfer is about forgetting how to decide.

A.7 Critical Dimensions

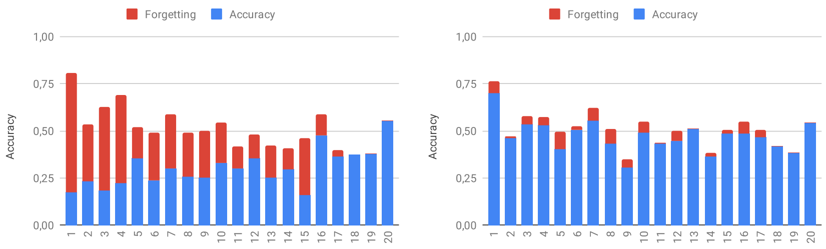
We also wanted to understand the importance of each dimension the masks were modulating. To test this, we take one of our trained models and selectively turn off (i.e. set to zero) the mask value associated with each activation map from the KB. We then analyze how many samples from the test set are incorrectly labelled compared to the regular model. We use this measure as a proxy for understanding the relevance of each activation map.

Table 4 shows that, for a given task, relatively few feature maps (on their own) have a critical impact on classification. However, a few feature maps show a disproportionate effect on classifying samples. These are feature maps that when turned off result in accuracy loss of 1% or more in task accuracy. We term these **critical dimensions**. Blocks closer to the output show an increase in the number of samples affected by a critical dimension. We hypothesize this is because layers closer to the output are believed to be more discriminative in nature, which would naturally lead to having more critical dimensions. This type of analysis, however, is limited in nature. We believe the impact of these dimensions should be studied not in isolation, but by their interactions with other dimensions. We have not found any tractable way of approaching such an analysis.

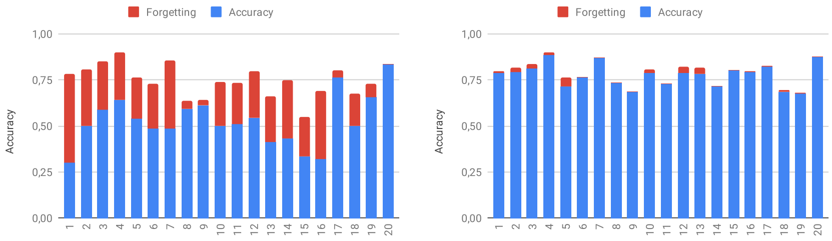
Table 4: The average percentage of dimensions that have a certain level of impact on performance for a model trained in 20-split CIFAR100. Results are averaged by the number of tasks. We see that most dimensions have no impact on samples, while a relative small part of them have relevant impact. Each column shows the percentage of dimensions that negatively impacts the accuracy by an X or greater percent for some task

	No impact	Impact	Critical						
Block	0%	>0%	>1%	>2%	>3%	>4%	>5%	>8%	>10%
Block 1	90.31	9.69	1.17	0.31	0.08	0.0	0.0	0.0	0.0
Block 2	95.0	5.0	1.52	1.25	1.09	0.98	0.86	0.78	0.66
Block 3	95.53	4.47	2.32	1.93	1.76	1.68	1.6	1.37	1.25

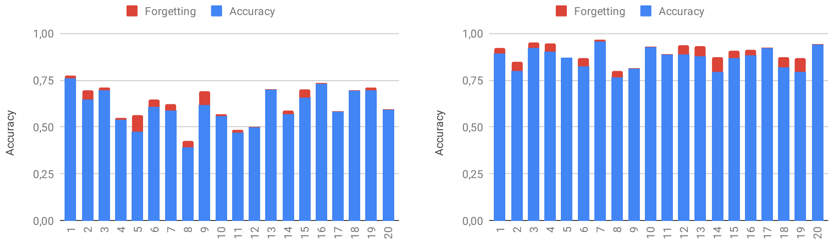
We then test how many of these critical dimensions are shared between tasks. We can see the results in Figure 8. A perfect KB would be composed of mostly reusable knowledge with some specific knowledge that is only relevant for a single task. What we see is that critical modules are shared between small groups of tasks (with one exception that is shared by 85% of the tasks). However, it is difficult to extrapolate conclusions over these findings. It is not clear how to quantify the ratio of specific knowledge to reusable knowledge, nor how to evaluate such a number if it was available.



(a) Shared (b) Shared + MetaLearning



(c) Shared + Masks (d) MARK + Task



(e) MARK + Random (f) MARK + Resnet

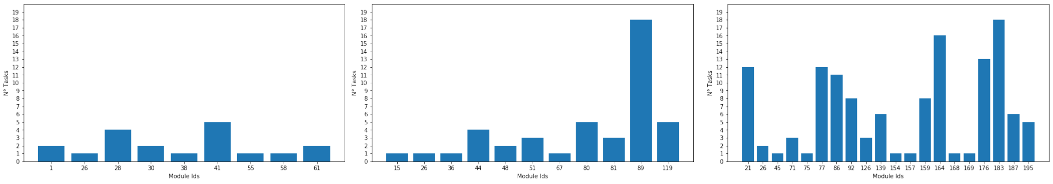
Figure 7: CF and accuracy for different versions of MARK tested on 20-Split CIFAR100. (a)Without using metalearning and mask-functions, performance is low and CF is high. (b) Adding only metalearning, performance is still low, but there is almost no forgetting. (c) Adding only mask-functions, performance increases but forgetting is still high. (d) MARK-Task, our full model,achieves high performance with almost no forgetting. (e) and (f) Quality of input features for mask-generating functions impacts overall accuracy but not forgetting. We hypothesize this is so because masks are acting as a self-attention mechanism that improves as richer features are provided.

A.8 Importance of Training Stages

We analyze the contribution of the different steps behind the training of MARK. As our method requires multiple steps in a particular order, it begs the question: are all of these steps necessary? To answer this, we compare two different versions of our model to MARK:

- **Feature:** This baseline is trained using only F^t with a classifier on top. It does not use KB, mask-functions M^t, or task specific classifiers C^t. This setup tests for no knowledge reuse between tasks.
- **No-Retraining:** We perform the KB-Querying phase of training only at the start of training a task. This setup tests the importance of retraining the mask functions and classifiers to take into consideration the new knowledge provided by the metalearning process.

Figure 9 shows the impact of each training method on task accuracy. We observe that, as expected, as we use more of MARK's components results are better. What is interesting is the difference between No Retraining and MARK at the beginning and by the end of training. For earlier tasks, not being able to capitalize on the full knowledge of the KB hurts the current's task accuracy greatly. However, as we near the end of our training, the difference between using the fully trained KB vs not using it



(a) Block 1 (64 modules)

(b) Block 2 (128 modules)

(c) Block 3 (256 modules)

Figure 8: Number of tasks that use a given critical dimension for each convolutional layer for a model trained on 20-split CIFAR100. Critical dimensions are defined as those that when turned off affect the correct classification of samples over a threshold. As we move closer to the output, layers increase the amount of critical modules, as well as intertask dependency on dimensions. However, no dimensions are universally useful. This suggests that while knowledge is reusable, there is still task-specific knowledge being stored in the KB. We use a threshold of N=5 samples which is a 1% deviation in accuracy.

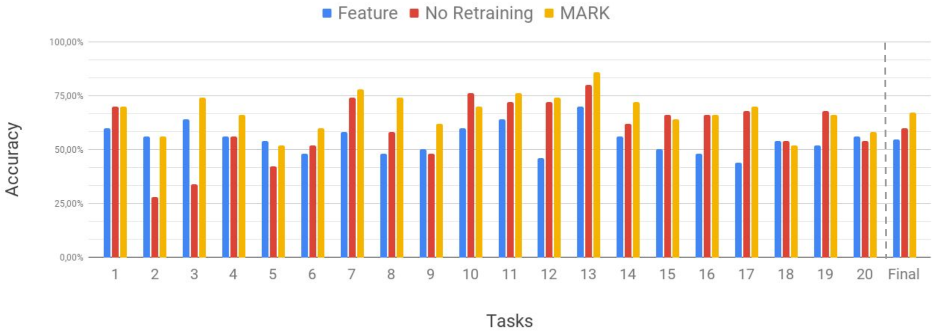


Figure 9: Accuracy in each of the 20 tasks in the 20-Split MiniImagenet sequence. First, we obtain the accuracy by only using the feature extractor. The No Retraining method shows a vast improvement on average, with MARK being the superior model. Note how the differences between No Retraning and MARK dwindle the more tasks we have trained. This suggests that MARK's KB is storing reusable knowledge that is available already at the start of training later tasks. The final three columns show the average of the complete sequence.

dwindles. We hypothesize this is because MARK's KB has already acquired sufficient knowledge from previous tasks, to perform well on the current task.

A.9 Number of Updated Parameters

In section 4.2.1 we mention a metric based on "number of updated parameters", however, we do not explain exactly what it means. We explain this now. For every task we compare the model before training on task t and after training. We compare the mean of the absolute difference between the *"before"* and *"after"* model against a threshold that is dependent on the standard deviation of the weights of the *"before"* model. If it's above this threshold we assume the weight was updated, else we don't. Standard deviations and means are calculated on a per-channel basis.

$$\sum_{i=1}^{n} \frac{|\theta^i_{before} - \theta^i_{after}|}{n} < \mu\sigma_{before}$$

NeurIPS Paper Checklist

1. For all authors...
    (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? Yes. For metalearning as key component for not forgetting, see section 4.2.3. For state of the art results, see Table 2. For reduced parameter use, see "Parameters" section in Appendix.
    (b) Did you describe the limitations of your work? Yes.
    (c) Did you discuss any potential negative societal impacts of your work? No. Our work is too broad and applicable for us to determine immediate applications that may be harmful to society.
    (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? Yes.
2. If you are including theoretical results...
    (a) Did you state the full set of assumptions of all theoretical results? n/a
    (b) Did you include complete proofs of all theoretical results? n/a
3. If you ran experiments...
    (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? Yes. We include a .zip file with our code, and we list some hyperparameters in the Experiments section and all of them in the appendix.
    (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? Yes. See the appendix.
    (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? Yes. We include standard deviations on all of our figures. We run experiments with 3 seeds.
    (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? Yes. See appendix for GPU details.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
    (a) If your work uses existing assets, did you cite the creators? Yes.
    (b) Did you mention the license of the assets? No. We use CIFAR-100 and MiniImageNet, very standard benchmarks.
    (c) Did you include any new assets either in the supplemental material or as a URL? No.
    (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? No. We use CIFAR-100 and MiniImageNet, very standard benchmarks.
    (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? No. We use CIFAR-100 and MiniImageNet, very standard benchmarks.
5. If you used crowdsourcing or conducted research with human subjects...
    (a) Did you include the full text of instructions given to participants and screenshots, if applicable? n/a
    (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? n/a
    (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? n/a