Reproducibility Challenge: Meta-Learning Representations for Continual Learning

Mihaela Georgieva Stoycheva
KTH Royal Institute of Technology
mgst@kth.se

Sergio Liberman Bronfman
KTH Royal Institute of Technology
liberman@kth.se

Konstantinos Saitas Zarkias
KTH Royal Institute of Technology
kosz@kth.se

Abstract

In this report we briefly introduce the *MRCL* method presented in Meta-Learning Representations for Continual Learning by Javed and White [2019] for learning representations which promote fast model adaptability to new tasks while trying to minimise catastrophic forgetting, which is the problem of continual learning. Additionally, we report results of our implementation based on the presented approach. Based on our level of knowledge in this topic, we found that the article is not reproducible on its own because several details essential for its implementation are not explicitly stated in the paper or are misleading. Finally, we discuss the issues and differences between expected and obtained results. Our implementation code can be found here: https://github.com/sergiolib/reproduce_oml using TensorFlow 2, while the original paper's code is written using PyTorch.

1 Introduction

Continual learning is based on the idea that an agent is being presented with continuous stream of data and its goal is to learn from it and adapt. The challenge comes from the fact that a model designed for this task has to exploit all the data to learn fast from few examples whereas at the same time it needs to retain all the previously acquired knowledge. This is a known shortcoming of deep neural networks, namely their inability to learn information continuously while keeping the knowledge from previous examples. The process of forgetting what has already been seen is also known as catastrophic forgetting and a lot of work has been done to alleviate this problem. The methods that deal with catastrophic forgetting can be divided into three main groups: generating samples from previous tasks, changing the online update to retain the knowledge and using sparse semi-distributed representations.

Meta learning, on the other hand, aims to find a good initialisation of a set of parameters based on a set of tasks related to, but not exactly the one that the model addresses. The idea is to be able to find the best parameters from a small amount of training data, given that a big amount of data from others tasks is available. Methods based on the work presented in Finn et al. [2017] are the current state of the art for Meta learning.

Meta-learning Representations for Continual Learning Javed and White [2019], or *MRCL*, tries to address Continual learning problems with Meta learning based solutions. The idea is to not find the best initial parameters, but rather find the best optimal trajectories of the parameters in their space in a Meta-learning fashion. The best optimal trajectories are defined either as perpendicular or parallel to all tasks. This way, in a perfectly pretrained model, every time a new task is learnt the previously

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

learnt ones are not forgotten. In practical terms, MRCL finds good representations for every related task with considerably few non zero representation terms so that only those related to the specific task get updated during the online updates and the rest remain untouched.

2 Method

The method Meta-learning Representations for Continual Learning consists of two sub-models, namely a representation learning network (RLN) and a task learning network (TLN). The network is set up so that it is possible to train only part of the parameters in each of the training steps using different objectives. Additionally, there are two sets of continual data, namely one used for pretraining and one used for online training. It is suggested to divide the pretraining dataset into two mutually disjoint sets $\mathcal{S}_{remember}$ and $\mathcal{S}_{learn}$ in order to stabilize the training procedure.

During the pretraining procedure the RLN aims at learning a good meta-learning representation that is later used for the online learning step. First, two datasets are constructed from the pretraining data. The first consists of a randomly sampled task from $\mathcal{S}_{remember}$ and is denoted as trajectory dataset $\mathcal{D}_{traj}$, while the second is a sequence from a random subset of tasks sampled from $\mathcal{S}_{learn}$ and is denominated as random dataset $\mathcal{D}_{rand}$. Both datasets correspond to sample sequences of the same tasks.

The task of finding the best representation is achieved by first making a copy of the initial weights of the TLN. Then, an inner loop that only updates the weights of the copied TLN is executed. This inner optimisation uses standard gradient descent and considers only the trajectory dataset $\mathcal{D}_{traj}$. After that the weights of both the RLN and the original TLN are updated using a concatenated sequence of the trajectory dataset and the random data $\mathcal{D}_{meta} = (\mathcal{D}_{rand} + \mathcal{D}_{traj})$. In this case the Adam optimiser is used and the loss is calculated using the output of the copied TLN. The pretraining procedure requires storing two versions of the TLN in memory.

Javed and White [2019] present a problem formulation of the continual learning prediction problem, that assumes the input consists of infinite samples in the form

$$(X_1, Y_1), (X_2, Y_2), ..., (X_t, Y_t), ... \tag{1}$$

where X_t is input and Y_t is the associated output for a specific pair and belong to sets $\mathcal{X}$ and $\mathcal{Y}$. The marginal distribution that defines how often a specific task is seen is defined by the density function $\mu : \mathcal{X} \to [0, \infty)$. The authors point out that assuming such distribution results in correlated sequences, a setup in which usual learning algorithms that work well on iid data would most likely fail.

With this notation in mind, the goal of continual learning is similar to most supervised classification problems, that is to find a function $f : \mathcal{X} \to \mathcal{Y}$ that can infer y from x. In the context of the method presented by Javed and White [2019] that is presented later in this report, the function f is parameterised with θ and W and is defined in Equation 2

$$f_{\theta,W}(x) = g_W(\phi_\theta(x)), \quad \phi_\theta(x) : \mathcal{X} \to \mathbb{R}^d, g_W(x) : \mathbb{R}^d \to \mathcal{Y} \tag{2}$$

In the equation above $\phi_\theta(x)$ is a shared representation function and $g_W(x)$ is task-specific. The objective, like in most supervised problems, is to minimise the expectation of the error $\ell(f(x), y)$, where $f(x)$ is the predicted label and y is the true label. The objective is defined as shown in Equation 3.

$$CLP(\theta, W) = \mathbb{E}[\ell(f(X), Y)] = \int \left[\int \ell(f_{\theta,W}(x), y) p(y|x) dy \right] \mu(x) dx \tag{3}$$

As mentioned, although the objective resembles the one of a standard supervised classification task, since the data might be correlated, the iid assumption fails in this setting. This makes the problem more difficult to solve with standard classification algorithms.

In order to tackle this issue, Javed and White [2019] propose the following objective of MRCL (OML) as defined in Equation 4.

$$\texttt{OML}(\theta, W) \overset{def}{=} \int \mathbb{E}\left[\texttt{CLP}\left(U(\theta, W, \{(X_{t+1}, Y_{t+1})\}_{i=1}^{k})\right) | X_t = x\right] \mu(x) dx. \tag{4}$$

For a set of different tasks (X_t, Y_t) and a parameter space (θ_t, W_t) at a given timestep, the update of the objective with stochastic gradient descent after k steps can be formulated as $U(\theta_t, W_t, \{(X_{t+1}, Y_{t+1})\}_{i=1}^{k})$ to produce (θ_{t+k}, W_{t+k}). In the context of OML, during the online updates, the θ parameter is fixed and only the W changes. The idea of this approach is that θ will be optimised for a better overall prediction accuracy but under the constraints of the online updates which is based on the representation of the meta-objective.

Once the model is pretrained, an online learning procedure is performed by freezing the RLN parameters and fine-tuning only the TLN using a training data stream of previously unseen examples. Later, an evaluation dataset should be reserved to evaluate the effectiveness of the training.

3 Datasets

In this article, two popular datasets for evaluating meta-learning algorithms are used. In this context, datasets that consist of multiple tasks, or classes, are more favourable in order to evaluate how well the algorithms can adapt to new information.

3.1 Incremental Sine Waves (ISW)

To evaluate the performance of *MRCL* in regression tasks, a set of randomly sampled sine waves varying in amplitude and phase were used. We followed closely the instructions of the paper to sample the data sequences and the network set up. Each of the sine functions has a one hot encoded identifier, a fixed phase ($\phi \sim U(0, \pi)$) and an amplitude ($A \sim U(0.1, 5.0)$). Its support is also randomly sampled ($z \sim U(-5, 5)$).

In each epoch, $K = 10$ functions were sub sampled. The identifiers for each of these functions were reassigned from $k = 0$ to $k = 9$ and assigned to each pair (z, y) where $y = sin_k(z)$. For each function, 40 sequences are sampled for the pretraining and 50 for the evaluation. To avoid excessive interference, the network is set up considering not only the support z but also the identifier k in the one hot encoding as inputs, and y as output.

3.2 Split-Omniglot

In the case of classification, the Omniglot [Lake et al., 2015] dataset was used. It consists of images illustrating hand-written characters from 50 alphabets. We use the standard split of 964[1] classes over 30 alphabets in the background dataset and 659 classes over 20 alphabets in the evaluation dataset. All classes contain 20 samples each. The background dataset is used entirely for pretraining in the context of MRCL, whereas the evaluation dataset is used for online training. Additionally, each class in the evaluation dataset is divided into 15 samples for training and 5 samples for testing.

In the context of the MRCL method the background dataset is split into two disjoint subsets, namely $\mathcal{S}_{learn}$ from which we sample $\mathcal{D}_{traj}$ and $\mathcal{S}_{remember}$ from which we sample $\mathcal{D}_{rand}$. The split is done by dividing the 964 classes in two where the first half is used for $\mathcal{S}_{learn}$ and the second one for $\mathcal{S}_{remember}$. For pretraining we randomly select a single class and we use all 20 samples for that class to create $\mathcal{D}_{traj}$. On the other hand, for constructing $\mathcal{D}_{rand}$ we choose 10 random classes and take one random sample from each class. Note that this strategy is not described in the original work by Javed and White [2019] but it is used in their official repository. For the online-training procedure we use the split of 15/5 of the evaluation dataset by selecting a random class and using the 15 samples to construct $\mathcal{D}_{traj}$ and 5 samples for $\mathcal{D}_{rand}$.

4 Baselines

The results that Javed and White [2019] present consist of the proposed method MRCL, compared with 4 other approaches, namely Scratch, Pretraining, SR-NN, Oracle. They test all these models on the two datasets mentioned above.

[1] In their work Javed and White [2019] claim there are 963 classes in the background set, which is incorrect.

4.1 Scratch

Description: This method is simply the MRCL algorithm without the pretraining procedure. Thus, the model starts with a random initialisation of the RLN and TLN network and tries to learn online class by class. **Expectations:** This is expected to perform quite poorly in any setting since the tasks are fed sequentially and there are only limited samples per task. More importantly, the network needs to be tuned in just one epoch with a small number of samples per class and keep learning new classes without forgetting the old ones. This methodology would require higher learning rates than the usual online training. For example, a bad performance is expected at the start of online training for Omniglot since the model will try to optimise for one class with 20 samples and then a new class will be introduced thus potentially overriding the progress of the network for the previous task learned.

4.2 Pretraining

Description: Another simplistic method they compare their results with is building a neural network with similar architecture trained with standard gradient descent. The data are shuffled and considered IID and this time the model does not have a meta-learning objective but simply tries to optimise for the loss of the mean-squared error (MSE) for regression or cross-entropy for classification. In the online training phase, the first few layers of the network are fixed and the rest are trained as normal. To find exactly *how many* layers to fix in online training, Javed and White [2019] do an architecture search by fixing different number of layers at each time, and picking the best one based on a validation set. [2] For the **regression task**, this network becomes a Multi-Layer Perceptron (MLP) with 300 units in each layer and a single node output layer to predict the next step of the function. All the other hyper-parameters are used as stated in Table 1 of Javed and White [2019][3]. For the **classification task**, doing an architecture search is more difficult since it consists of convolutional layers which need to be initialised with a specific number of filters and output size. Since Javed and White [2019] do not mention any information on how this architecture search was carried out we used the standard architecture of MRCL. **Expectations:** Although slightly better than the scratch method described earlier, the pretraining baseline should behave well when the amount of learnt classes is low, but it should be rapidly overthrown by MRCL as the ability of learning new classes would cause interference with already seen ones.

4.3 SR-NN

Description: This is a method described by Liu et al. [2019] which aims at learning a sparse representation to achieve a better and more stable performance on the objective. Additionally, an extra hyper-parameter β can control how sparse the representation of the network should be when training. In their work Javed and White [2019] mention that they do a search for the best value of β and report the results of that but without stating how this search was done or which was the actual best β value. **Expectations:** In the original paper of SR-NN, the method is just used for reinforcement learning tasks so there is no clear expectation on how it should perform in regression or classification tasks.

4.4 Oracle

Description: Oracle is a method name used in machine learning to represent the highest possible performance in optimal settings. **Expectations:** Javed and White [2019] share that Oracle is pre-trained with independent and identically distributed (IID) data, but they do not explain in details how the method is implemented. Thus, we assume that the Oracle method is following exactly the same training procedure as MRCL, as presented in Section 2, but with the only difference on how the data are fed to the network. However, this implementation did not give results worth presenting in this report as they significantly deviated from the ones reported in Javed and White [2019]. The achieved performance was slightly better than random.

[2] Note that they do not mention how extensive this architecture search is or what the validation set consists of.

[3] Many parameters did not appear at all in that table, and thus had to be assumed. These included initialisation methods, amount of epochs and regularisation.

5 Results

In this section we report the results we achieved with the subset of experiments we performed from Javed and White [2019] and point out the observations we found. Further comparison and comments on the similarities or discrepancies between the results in Javed and White [2019] and our results is reported in section 6.

5.1 Experiments

In the first version of their original work Javed and White [2019] present the results from multiple experiments that evaluate the performance of MRCL on both regression and classification tasks and compare it to a combination of MRCL with other continual learning approaches. However, we decided to reproduce the results of a subset of those experiments due to time constraints. Following is a description of this subset, as well as a motivation for our choice.

In the original report, Section 4.4 firstly introduces figure 3a for ISW in which the mean squared error (MSE) for MRCL and the baselines (pretraining, SR-NN and oracle) are computed for different number of functions learned in the tasks training. Figure 3b shows the MSE that every task ID got after training for all the functions on two of the baselines (pretraining and SR-NN). From figure 4a, the section shows for Omniglot the training set accuracy vs. the number of classes learned for four baselines (scratch, pretraining, SR-NN and oracle). On figure 4b, they report similar results for the test set accuracy. Further on, Section 4.5 reports the quality of the representations learnt as sparse vectors with the most uniform distribution over all the tasks. These are computed just for Omniglot and depicted on Figure 5.

In Section 5 the authors report their integration of MRCL with baselines of continual learning techniques. These baselines include (1) Fully online SGD updates one point at a time in the order of the trajectory, (2) Approximate IID training / SGD updates on a random shuffling of the trajectory for removing correlation, (3) ER-Reservoir (Chaudhry et al. [2019]), (4) MER (Riemer et al. [2019]) and (5) EWC (Lee et al. [2017]).

For each of these the authors make 2 setups using the Omniglot dataset. In the first one they run 50 classes in a one-class-per-task fashion, while in the second they set up 20 tasks in a five-classes-per-task way. For each of these set ups, the authors report 3 experiments: one with standard training and testing, another using MRCL and a third one using the pretraining baseline.

Our initial goal was to try and reproduce all of the experiments. Due to complications of reproducing the proposed MRCL method, we prioritised to focus on reproducing the basic experiments of Section 4. As the described methodology was not completely and successfully re-implemented, the experiments in Section 4.4 and Section 4.5 were not being reproduced correctly and thus, the experiments of section 5 were not addressed.

An observation we can do regarding the omitted experiments from Section 5 is that the authors do not provide a reference or code in their repository to reproduce these values. A researcher has asked the first author for the implementation of this part of the paper through the issues system in his Github repository in July 21, 2019.[4]

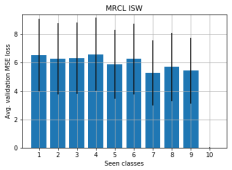
From the experiments, the results shown in Section 4.4 for Omniglot are replaced in the newer version of the paper, released in late October 2019, by different and slightly worse values. Also, another test is introduced to the Figure 4 which shows the difference in errors between a single pass training (as it was assumed to be conducted before) and a multiple-pass problem. Also, the previous Figure 5 was moved into a Figure 6 and the new Figure 5 shows results of tests perpetrated on the MiniImagenet dataset.

5.2 ISW

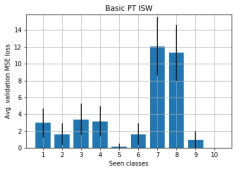
The results obtained on the figures 1 are not a reproduction of those in Javed and White [2019]. The errors represented are, in average 12 times larger and their standard deviation is similarly increased. The errors obtained for validating the Pretraining Baseline were also very uneven compared to those of figure 3b. Despite of that, the main results commented in that article can be observed in the

[4]The issue can be found in the link https://github.com/khurramjaved96/mrcl/issues/1

(a) Mean of MSE loss of MRCL and Pretraining Baseline over 50 evaluations on different tasks. Each time a full class was trained, it was immediately tested on unseen data.

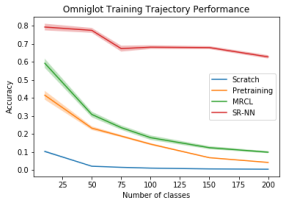


(b) MSE loss for testing on MRCL only data classes learnt so far during online training on unseen data. Error bars show 95% confidence after 50 runs on different tasks.

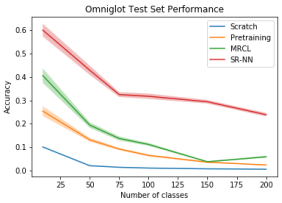


(c) MSE loss for testing on Pretraining Baseline only data classes learnt so far during online training on unseen data. Error bars show 95% confidence after 50 runs on different tasks.

Figure 1: Although the losses achieved are higher than in the paper, the experiments the main points of it: the lower MSE errors on Baseline Pretraining are always the last seen classes, and the error of classes seen before increases.



(a) Omniglot training accuracy



(b) Omniglot testing accuracy

Figure 2: Comparison of the methods in the online training and testing accuracy

plots. The last classes learnt have a lower error than those learnt previously, and MRCL shows great resilience to forgetting the first tasks.

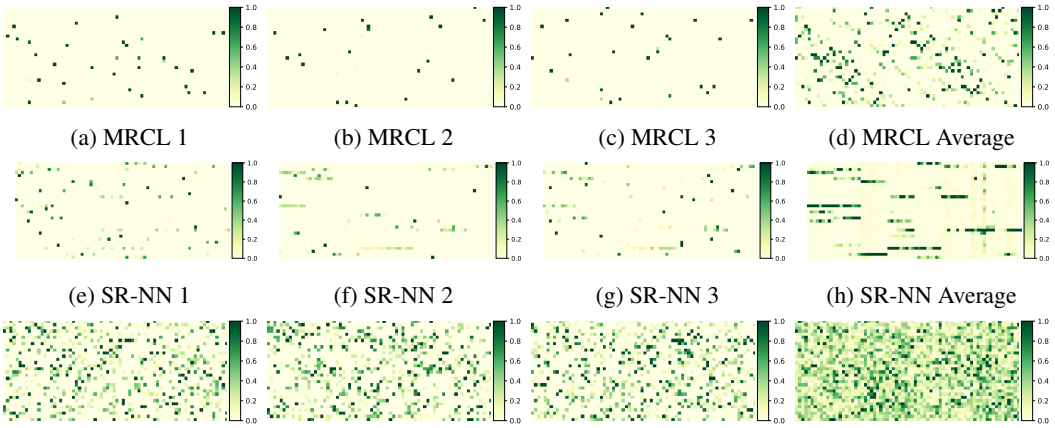
Figure 2a shows also that MRCL achieves a lower error per class than that of Basic Pretraining. In this plot, the SR-NN method was omitted because the evaluation of the sole MRCL was prioritised. Also, the authors do not provide code for testing SR-NN against MRCL as performed in the original figure 3 of Javed and White [2019] which means that we do not have the β parameter used in plotting this figure.

5.3 Omniglot

Similarly, for the Omniglot experiment, we did not manage to reproduce the performance of MRCL in comparison with the other methods. Specifically, as seen in figures 2, both the online training accuracy and the online testing accuracy of SR-NN outperforms all the other methods. With the exception of MRCL, the performance of the compared methods is similar to what is reported by Javed and White [2019] but slightly worse. This might be due to implementation differences for which we report in section 6. An important note here is that the fact that the performance of our implementation of the basic pretraining approach is comparable by the one achieved in the original work makes us believe that our online training procedure is correct. Thus, we speculate that the learned representation for MRCL is incorrect and this could be due to number of implementation details that are missing from the description in the original paper. For example, when choosing the random dataset $\mathcal{D}_{rand}$ it is not specified how large is it or how should it be constructed. Additionally, it is not mentioned for how long the network should be pretrained and if there use any method to detect over-fitting.

Figure 3 illustrates the learned representation for 3 randomly selected samples and the average activations for the whole Omniglot dataset for the MRCL, Pretraining and SR-NN method. The representations have length 2304 and similarly to the original paper we reshape them into 32x72 plots. The results for the three random samples for MRCL are comparable to the ones presented in the work

of Javed and White [2019] where they report high sparsity on the learned representations. However, the average activations over the background and evaluation sets for MRCL is contradictory to the original results. In the plot 3d we can see that there is a high number of dead neurons, whereas in the results presented by Javed and White [2019] they achieve a representation without dead neurons. This result supports our claim that the reimplementation of the representation learning method is incorrect. On the other hand, the representations learned by SR-NN are less sparse then the ones acquired by MRCL. Additionally the activations that are reported for the Pretraining model are considerably more dense then the results of the other approaches. More specifically, we report an average activation sparsity of 98% for MRCL, 95% for SR-NN and 70% for Pretraining.



(a) MRCL 1 (b) MRCL 2 (c) MRCL 3 (d) MRCL Average

(e) SR-NN 1 (f) SR-NN 2 (g) SR-NN 3 (h) SR-NN Average

(i) Pretraining 1 (j) Pretraining 2 (k) Pretraining 3 (l) Pretraining Average

Figure 3: Learned representations by the MRCL, SR-NN and Pretraining models for three random samples.

6 Discussion

There were multiple omissions or vague statements in the paper that made the reimplementation of the method a difficult task. Firstly, we tried reproducing the results presented in the paper only with the knowledge and information provided in the work but soon a lot of questions arose. In order to solve these issues we made some assumptions and at times we had to inspect the authors' code.

6.1 Dataset issues

Additional to the issues already presented, there are problems related to the used datasets that should have been justified and corrected by the authors in order to make the method reproducible.

ISW: Although not specified in the paper, the ISW dataset should contain a certain number of pairs (x, y) for each of the samples functions and for each of the sampled repetitions of pretraining. This is exactly the case of the official code repository, in which the authors generate 32 samples per function repetition. This higher number allows the pretraining to converge faster to a good result as a higher amount of data available for pretraining allows learning better representations for similar tasks, but can also make the network learn correlations in an epoch that are harmful for the actual learning of representations for Continual Learning.

Omniglot: In their paper they do not state the pretraining procedure they followed for the methods of Pretraining and SR-NN, other that the data are shuffled and IID. After a quick inspection on their published code, we observed that they did not use the same amount of pretraining data as they did for their own MRCL method. Specifically, we report that for the Pretraining and the SR-NN methods they train with 900 classes instead of 964 (as MRCL) and instead of 20 samples per class, they use 15. This means that the MRCL method is pretrained with 964 * 20 = 19.280 samples whereas the Pretraining and the SR-NN are pretrained with 900 * 15 = 13.500 samples. This difference can significantly decrease the evaluation performance of these two models.

6.2 Hyper-parameter omissions

When presenting the setup or reporting results for MRCL and all the baseline methods, Javed and White [2019] do not report all the hyper-parameters used or they contradict previous statements. For example, they do not mention the amount of passes through the meta learning loop and in the online training of MRCL. After some empirical experimentation we choose the amount of epochs in meta learning to be 20000 and just one pass over the training data on the online training. Additionally, the number of epochs for the compared methods is not mentioned. Thus, after a hyper-parameter search for different number of epochs we choose 30 for SR-NN, 1000 for basic pretraining on Omniglot and 40000 for basic pretraining on MRCL.

For the networks used in the Omniglot dataset, Table 3 in the original paper reports the number RLN and TLN layers and how many of convolutional kernels in each RLN layer but omits the number of units in the fully connected layers [5]. Moreover, when the architecture used for MRCL is described for the ISW dataset in section 4.2, they state "We use six layers for the RLN and two layers for the TLN." but in their appendix in Table 2 which contains the hyper-parameter values of MRCL for the ISW dataset they write "Total layers in the fully connected NN = 9". For this issue we assumed that 6 layers would go to the RLN, 2 hidden ones to the TLN and a final one for the output.

The best β value of SR-NN, that is, the one that gave the highest accuracy is not reported in the original work. They do report they did a hyper-parameter search to find the best value but do not explicitly mention what was that value or how extensive the search was. Instead, the value $\beta = 0.05$ is reported as the one that gave the lowest sparsity level in their figure 4. To deal with this, we did our own hyper-parameter search for β (see in Appendix) and we report that the value that gave the highest accuracy was $\beta = 0.1$. In the pretraining baseline method, no details are given about the architecture search setup. This includes no mentioning of how many layers and nodes are tested. Nor is mentioned how they make the validation set for the search. After exploring a few combinations of hyperparameters in a grid search, the preliminary results obtained suggested that using the same architecture as in MRCL was in fact optimal. For the learning rates we search which one minimises the validation loss. In the case of ISW, the best LR was 1e-06, while in Omniglot it was 1e-03. For the validation set in Omniglot, the split was done as in the pretraining: leaving 5 samples per class for validation, and the rest in training. As for the ISW task the split for MRCL is done as in pretraining where we generate 40 times 10 functions of samples of length 32 for the training set. The validation set is constructed by sampling the same 10 functions only 32 samples for each.

6.3 Method description and code mismatches

We found several differences between the settings stated by Javed and White [2019] and those used in the official code. Most of these settings have never been modified in the official repository's history and thus we can assume that these differences have been introduced since the design of the method. One example is that the representation length of ISW is not mentioned, although the official code repository assumes a representation of 900 outputs. This number is understood as the total amount of different functions that the network will end up seeing and learning from, and thus a perfect representation should average across all the classes a uniformly dense area. This issue is more profound when observing that in Table 1 the amount of layers and width of each layer is reported, thus one can imply that the representation has the same width as the rest of the layers as well. This case was initially considered and the results were incorrect.

6.4 Other issues

In Section 4.1 of the original paper where the Omniglot dataset is described, the authors mention "Moreover, we use the "single-pass through the data" protocol used by Lopez-Paz and Ranzato [2017]." However, in Lopez-Paz and Ranzato [2017] there is no such protocol mentioned. Additionally, they do not report any results achieved by the Scratch baseline method for the ISW dataset, and no justification is done on this matter. We can only presume that the results were not satisfactory and worth presenting. Finally, in Section 5 of their work the authors mention they compare their results with a method called EWC and they cite the paper Lee et al. [2017] which is not related to EWC but proposes another method called IMM.

[5] In their official code they use 1024 and 1000 units for the 1st and 2nd layer in the TLN respectively

References

A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato. Continual learning with tiny episodic memories. *arXiv preprint arXiv:1902.10486*, 2019.

C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

K. Javed and M. White. Meta-learning representations for continual learning. *NIPS*, abs/1905.12588, 2019.

B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. doi: 10.1126/science.aab3050.

S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in neural information processing systems*, pages 4652–4662, 2017.

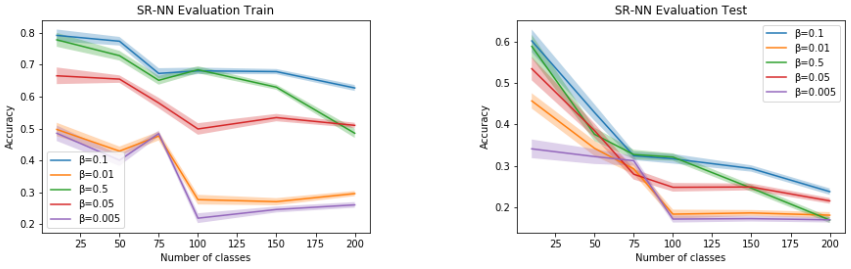
V. Liu, R. Kumaraswamy, L. Le, and M. White. The utility of sparse representations for control in reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4384–4391, 2019.

D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017.

M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, , and G. Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2019.

A Additional Experiments

As mentioned in 6, the authors do not report what type of hyper-parameter search they implemented to find the best β value for SR-NN so we did a search with the values $\beta = [0.5, 0.1, 0.05, 0.01, 0.005]$ and report the results in figure 4 and the best value $\beta = 0.1$.



(a) Training during evaluation

(b) Testing during evaluation

Figure 4: Results for the β value search.