Understanding Catastrophic Forgetting and Remembering in Continual Learning with Optimal Relevance Mapping

Anonymous Author(s)
Affiliation
Address
email

Abstract

Catastrophic forgetting in neural networks is a significant problem for continual learning. A majority of the current methods replay previous data during training, which violates the constraints of an *strict* continual learning setup. Additionally, current approaches that deal with forgetting ignore the problem of catastrophic remembering, i.e. the worsening ability to discriminate between data from different tasks. In our work, we introduce Relevance Mapping Networks (RMNs). The mappings reflect the relevance of the weights for the task at hand by assigning large weights to essential parameters. We show that RMNs learn an optimized representational overlap that overcomes the twin problem of catastrophic forgetting and remembering. Our approach achieves state-of-the-art performance across many common continual learning benchmarks, even significantly outperforming data replay methods while not violating the constraints for a *strict* continual learning setup. Moreover, RMNs retain the ability to discriminate between old and new tasks in an unsupervised manner, thus proving their resilience against catastrophic remembering.

1 Introduction

Continual learning (*CL*) refers to a learning paradigm where different data and tasks are presented to the model in a sequential manner, akin to what humans usually encounter. But, unlike humans or animal learning, which is largely incremental and sequential in nature, artificial neural networks (*ANNs*) prefer learning in a more concurrent way and have been shown to forget *catastrophically*. The term *catastrophic forgetting (CF)* in neural networks is usually used to define the inability of *ANNs* to retain old information in the presence of new one.

Given the complex nature of the problem, most *CL* methods understandably relax the constraints of what we term as a ***strict CL*** setup (Section 3.1), i.e. roughly, not using any (saved or generated) negative exemplars[6, 36] and largely preserving the base neural model. Practically, such a paradigm may arise in certain edge devices and resource constrained remote systems, and theoretically, such a setup is important (as we will show) in building a better understanding of the inner workings of neural continual learning. Therefore, in this work, we primarily focus on studying the *strict CL* setup.

Catastrophic Forgetting is a direct implication of *CL* in *ANNs* and is largely considered a direct consequence of the overlap of distributed representations in the network. Most prior works deal with *CF* by either completely removing the representational overlap [11, 20] or more frequently, by replaying data from previous tasks. Since Robins [43] showed the promise of memory replay methods in dealing with *CF* and the prevalence of cognitive/neuro science inspired theories regarding memory, it is of no surprise that *rehearsal,memory buffer or generative replay, etc.* methods dominate

Submitted to the 5th Workshop on Meta-Learning at NeurIPS 2021, held virtually. Do not distribute.

the current state of the art (SOTA) benchmarks [50, 17, 40, 28]. However, they clearly *violate* the conditions of the *strict CL* paradigm. Crucially, data replay methodology in its effort to deal with *CF* often leads to a reduced capability of the network to discriminate between old and new inputs [47]. This is referred to as **Catastrophic Remembering (CR)** (refer to Section 3.2) and has been shown to be a significant limitation of replay methods [43, 47].

The goal of this work is to develop a method for continual learning for deep neural networks which can alleviate the twin problem of Catastrophic Forgetting and Catastrophic Remembering at the same time, without violating or relaxing the conditions of a *strict* continual learning framework.

Our proposed approach builds on the following **Optimal Overlap Hypothesis**: *For a strictly continually trained deep neural network, catastrophic forgetting and remembering can be minimized, without additional memory or data, by learning optimal representational overlap, such that the representational overlap is reduced for unrelated tasks and increased for tasks that are similar.*

Inspired by this hypothesis, we propose *Relevance Mapping* for continual learning to alleviate *CF* and *CR*. During the continual learning process, our method learns the neural network parameters and a task-based relevance mask on the hidden layer representation concurrently. The binary relevance mask keeps a portion of the neural network weights static and hence is able to maintain the knowledge acquired from previous tasks, while the rest of the network adapts to the new task. Our experiments demonstrate that Relevance Mapping Networks outperform related works by a wide margin on many popular continual learning benchmarks (Imagenet-50, Permuted MNIST, Split MNIST, Split Omniglot, Split CIFAR-100), hence alleviating catastrophic forgetting without relaxing or violating the conditions of a *strict* continual learning framework. Moreover, we demonstrate that Relevance Mapping Networks are able to detect new sequential tasks in an unsupervised manner with high accuracy, as well as sustain the task separation without supervision by creating unique overlapping subnetworks, hence alleviating catastrophic remembering.

In summary, our contributions are:

- We introduce Relevance Mapping Networks which learns binary relevance mappings on the weights of the neural network concurrently to every task. We demonstrate that our model efficiently deal with the twin problem of catastrophic forgetting and remembering.
- We explore the concept of *Catastrophic Remembering* for deep neural networks and show that our method is capable of dealing with the same (becoming the first methodology to elevate catastrophic forgetting and remembering concurrently).
- **Our method achieves *SOTA* results on many popular continual learning benchmarks** under strict CL constraints and achieves better results than many other previous SOTA works which are not be defined under the *strict CL* setup.

2 Related Work

Continual Learning. Current continual learning mechanisms dealing with *CF* are broadly classified into *regularization approaches, dynamic architecture, complementary learning systems and replay architectures* [41]. Primarily based on the *Stability-Plasticity Dilemma* [35] concept, regularization approaches impose constraints on weight updates to alleviate catastrophic forgetting like *Elastic Weight Consolidation* (EWC) [20] and *Learning Without Forgetting* [32]. These methods do not ordinarily violate the conditions of a *strict CL* framework but have been shown to suffer from brittleness due to representational drift [50, 18] and thus are usually combined with other methods. Rehearsal/replay buffer methods, like [50] which are the state-of-the-art methods, use a memory store of past observations to remember previous tasks in order to alleviate the brittleness problem. However, these are not representative of *strict sequential learning* insofar that they still require re-learning of old data to some extent and perform significantly worse the less samples are replayed, and they may struggle to represent uncertainty about unknown functions.

There are no known modern methods which deal with *CR* in a *CL* framework, with our method being the *first of its kind* to be able to combat *CF and CR* in a *strict* continual learning framework.

Catastrophic Remembering refers to the tendency of artificial neural networks to abruptly lose the ability to discriminate between old and new data/task during sequential learning. It is an important problem and inherently attached to the problem of catastrophic forgetting. But, unlike the problem

of catastrophic forgetting, which has a rich literature of research, catastrophic remembering has not been explored outside of minor discussions in early works [46, 30, 11]. In this work, we discuss CR from a probabilistic perspective (Section 3.2) and demonstrate that related work suffers from CR in our experiments in Section 5.2. Finally, we demonstrate that our proposed Relevance Masking Networks are much more resilient to catastrophic remembering.

3 Strict Continual Learning and a Catastrophic Memory

3.1 Strict Continual Learning in Neural Networks.

This formulation of continual learning refers to a learning paradigm where *ANNs* are trained *strictly* sequentially on different data and tasks [6, 36]. Given a neural network f with parameters θ and $\mathcal{T}_i$ being the time interval of the training with task data $\mathcal{D}_i$ where i is the current task and $\mathcal{N}$ is the total number of tasks. The important conditions of the training paradigm are:

1. Sequential Training i.e. $\mathcal{T}_1[f_\theta(\mathcal{D}_1)] < \mathcal{T}_2[f_\theta(\mathcal{D}_2)] < ... < \mathcal{T}_\mathcal{N}[f_\theta(\mathcal{D}_\mathcal{N})]$ with the *single* f.
2. No negative exemplars, examples or feedback i.e. future (or past) data samples cannot be provided to the network with the current data/task in any way.
   $(\mathcal{D}_1 \cap \mathcal{D}_T) \cup ... \cup (\mathcal{D}_{T-1} \cap \mathcal{D}_T) \cup (\mathcal{D}_T \cap \mathcal{D}_{T+1}) \cup (\mathcal{D}_T \cap \mathcal{D}_{T+2})... \cup (\mathcal{D}_T \cap \mathcal{D}_{T+N}) = \emptyset$

As can be garnered from the above definition, any type of replay or buffer methods relaxes the above conditions. Many *CL* methods [45, 28, 13, 51] also change the *ANN*(f_θ) altogether by adding new convolutional/linear layers for each task (for e.g. using multi heads -different last linear layer for each task- has become a common practice), using a mixture of *ANNs*, using additional models (like generators) which relaxes the above conditions as well since we are not using the same *ANN* anymore. *RMNs* only use the original set of convolutional/linear weights of the base model for all tasks.

Catastrophic forgetting from a probabilistic view. Intuitively, given a learnt initial set of parameters θ_i for a neural network f and a task i with data D_i, the network's parameters get overwritten when it learns a new set of network parameters θ_{i+1} from new data D_{i+1} for the $(i+1)_{th}$ task. To facilitate the conceptual understanding of CF, we consider continual learning from a probabilistic perspective, where optimizing the parameters of $f_{\theta_{1:\mathcal{N}}}$ is tantamount to finding their most probable values given some total data $\{\mathcal{D} \mid \mathcal{D} \supset D_1, ..., D_n\}$ [20]. We can compute the conditional probability of the first task $\mathcal{P}(\theta_1|D_1)$ from the prior probability of the parameters $\mathcal{P}(\theta_1)$ and the probability of the data $\mathcal{P}(D_1|\theta_1)$ by using Bayes' rule. Hence, for the first task,

$$\log \mathcal{P}(\theta_1|D_1) = \log \mathcal{P}(D_1|\theta_1) + \log \mathcal{P}(\theta_1) - \log \mathcal{P}(D_1). \tag{1}$$

Note that the likelihood term $\log \mathcal{P}(D_1|\theta_1)$ simply represents the negative of the loss function for the problem at hand [20]. Additionally, the posterior term is usually intractable and only approximated for *ANNs* [50, 37, 20] and we are just considering it here without change for analysis purposes only. If we were to now train the same network for a second task, the posterior from (1) now becomes a prior for the new posterior. If no regularization or other method is included to preserve the prior information, we'd optimize for the second task,

$$\begin{aligned}\log \mathcal{P}(\theta_{1:2}|D_{1:2}) &= \log \mathcal{P}(D_2|\theta_2) + \log \mathcal{P}(\theta_1|D_1) - \log \mathcal{P}(D_2) \\ &= \log \mathcal{P}(D_2|\theta_2) + \log \mathcal{P}(D_1|\theta_1) + \log \mathcal{P}(\theta_1) - \log \mathcal{P}(D_1) - \log \mathcal{P}(D_2).\end{aligned} \tag{2}$$

If the likelihood term is not optimised over both θ_1 and θ_2, as would happen normally in a ordinary *ANN* training setup, the prior information can be overwritten, leading to the condition commonly referred to as catastrophic forgetting.

Overcoming catastrophic forgetting. We can clearly see from Eq. 2, if we had access to the previous data D_1 or if both θ_1 and θ_2 were independent of one another, we could approximate a well optimized posterior. However, even in a typical continual learning setup we cannot trivially assume access to the previous data or make independence assumptions on the sequentially learnt parameters. This, however, gives us a crucial conceptual insight into how to deal with *CF* and the mechanisms of current popular *CL* methods, which aim to overcome CF by relaxing either of the two restrictions. In particular, some recent works [34, 45, 16] try to effectively separate the model parameters θ_i for different tasks, as initially proposed by [10]. The most successful recent works [40, 50, 5, 13, 17] involve data replay methods, which relax the previous data availability restriction. Despite some success in dealing with CF, data replay drastically diminishes the discriminative ability of the *ANN*, a

phenomenon referred to as *Catastrophic Remembering* [43]. This usually happens because the *ANN* tries to learns a more general function $f(\theta_{1:\mathcal{N}})$ than necessary since now previous data is available, generalizing not only to the individual tasks, but to the entire sequential set of tasks $\{\theta \mid \forall \theta_i \subsetneq \theta_{1:\mathcal{N}}\}$ which has been referred to as *overgeneralization* [43]. The network can then experience a sense of *extreme deja vu* [47] and is unable to differ the old from new data or discriminate between tasks. Alongwith real world scenarios, this also underlines the necessity of a strict *CL* setup - which encourages study of effects of *CL* in *ANNs* by avoiding side-stepping the problem.

3.2 Catastrophic Remembering

For a better understanding of *CR* and why *CF* alleviation aggravates it, we calculate the posterior after the n_{th} task learnt continually using Eq. (2),

$$\log \mathcal{P}(\theta_{1:n}|D_{1:n}) = \log \mathcal{P}(D_n|\theta_n) + \sum_{i=1}^{n-1} \log \mathcal{P}(D_i|\theta_i) + \log \mathcal{P}(\theta_1) - \mathcal{C} \tag{3}$$

where $\mathcal{C}$ is a constant representing the sum of the normalization constants $\sum_{i=1}^{n} \log \mathcal{P}(D_i)$. As discussed earlier, the information of from the previous tasks is passed to the next sequential task as a prior ($\sum_{i=1}^{n-1} \log \mathcal{P}(D_i|\theta_i)$). The problem of loss of discriminative ability arises when for an arbitrary large n when the prior term far exceeds the currently optimized likelihood. For Eq. (3), that means

$$\log \mathcal{P}(D_n|\theta_n) << \sum_{i=1}^{n-1} \log \mathcal{P}(D_i|\theta_i). \tag{4}$$

In the context of data replay methods this intuitively means that if the number of data from the previous tasks $\{D_1, \ldots, D_{n-1}\}$ is far bigger than the data in the current task D_n the contribution of the present likelihood to the posterior is negligible and no new features are learnt by the *ANN* to account for the new dataset/task. This, in turn, gives the model a sense of false *familiarity* with a new input and the model is no longer able to discriminate between old and new inputs. The above explanation, though not exhaustive, provides a initial understanding from a Bayesian viewpoint.

A parochial outlook on generalization without general robustness may allow us to argue against the necessity of the *discriminative memory* property that *CR* attacks in *ANNs*. However properties like novel input and task detection are crucial in fields like AI, Computer Vision and Robotics. It can be necessary to detect new inputs to learn more robust features for the current data, e.g. a self driving network may need to identify whether it is familiar with a current set of input data for varied reasons. The loss of ability to differentiate amongst the already learnt tasks (also caused by *CR*) will cause perpetual source ambiguity in *CL* neural systems during inference which could be an undesirable loss of information. *Recognition and discrimination memory* are important aspects of human memory and learning - which is the inspiration for the field of *CL* - concepts which *ANNs* have been trying to replicate. However, as we have shown above, many *CL* methods can not robustly replicate this property even if they can alleviate *CF* in *CL*. This underlines the importance of study of *CR* in *CL* which may lead us to have a better understanding of robust *CL* systems in general.

4 Relevance Mapping for Continual Learning

We introduce *Relevance Mapping*, which is a method inspired by the *Optimal Overlap Hypothesis*, that aims to learn an optimal representational overlap, such that unrelated tasks use different network parameters, while allowing similar tasks to have a representational overlap. Note that our method avoids data replay and instead aims to achieve independence between network weights that are used for different sequential tasks.

Algorithmic implementation of Relevance Mapping. To illustrate and motivate Relevance Mapping Networks (RMNs) using a simple example, we consider a multilayer perceptron (MLP) with two layers f defined as $f(x) \triangleq \sigma(W_2\sigma(W_1x))$, where $x \in \mathbb{R}^{d_1}$, $W_1 \in \mathbb{R}^{d_2 \times d_1}$, and $W_2 \in \mathbb{R}^{d_3 \times d_2}$, and σ denotes a nonlinear activation function. We denote the set of weights as $\mathbf{W} \triangleq \{W_1, W_2\}$. Although it may depend on the dimensionality of the task, overparameterization occurs even in these simple MLP settings. For a sufficiently simple task, only a subset of the parameters in $\mathbf{W}$ are often required [9]. For example, if the optimization task has ground truth outputs specified as

$f^*(x) = \sigma(W_2^* \sigma(W_1^* x))$ for optimized weights $\{W_1^*, W_2^*\}$, and $||W_1^*||_0 + ||W_2^*||_0 \ll d_3 d_2 + d_2 d_1$ (i.e. the number of non-zero weights needed for the ground-truth function is much less than the number of total weight parameters) then only $||W_1^*||_0 + ||W_2^*||_0$ weight parameters are necessary to be learned in network f. In theory, if we could learn the *importance* or *relevance* of each weight node, we could apply a zero-mask to the non-essential parameters without pruning or modifying them and still successfully learn the ground-truth. A set of mappings can be denoted as $\mathbb{M}_{\mathbb{P}} = \{\mathbb{M}_{\mathbb{P}_1}, \mathbb{M}_{\mathbb{P}_2}\}$, where $\mathbb{M}_{\mathbb{P}_1} \in \{0,1\}^{d_2 \times d_1}$ and $\mathbb{M}_{\mathbb{P}_2} \in \{0,1\}^{d_3 \times d_2}$, explicitly representing the neuron-to-neuron connections of the network. The initialized relevance mappings of an *ANN* can be approximated by a logit-normal distribution mixture which is rounded during inference.

$$\mathbb{M}_{\mathbb{P}_i} \approx \prod_i \mathcal{L}_R(\mathcal{N}(\mu_i,\, \sigma_i^2)) \quad \text{and} \quad \mathcal{L}_R(x_i; \beta) = \frac{1}{1 + \exp(-(\beta(x_i - 0.5)))} \tag{5}$$

where μ, σ are the initializing distribution parameters and $\mathcal{L}_R$ sigmoidal pseudo-round function. This is done in order to make the mappings differentiable and the individual mixture components are jointly optimized for the task with the network parameters. In theory, *any* network f with weight tensors $\mathbf{W}$ can have such corresponding sets of neuron connection representations $\mathbb{M}_{\mathbb{P}_{1:T}}$ for T tasks/mappings, where each set $\mathbb{M}_{\mathbb{P}_i}$ activates a subnetwork mapping in f that could be used for various purposes for a task i. This simple example illustrates the trivial result that the over-parameterization property of modern networks[9].

Note that $\lim_{\beta \to \infty}(\mathcal{L}_R(x, \beta))$ for $x \in [0, 1]$ is equivalent to the rounding function. Here, β is a learnable, layer-wise parameter (i.e., in our implementation, there is one specific β for every layer of a given network) that controls the "tightness" of $\mathcal{L}_R$. To achieve an approximate neuron-connection representation, we define $\mathbb{M}_{\mathbb{P}} = \mathcal{L}_R(x; \beta)$ where x is initialized from some distribution with support $(0, 1]$ (in experiments, we initialize x with a clipped, skewed normal distribution). During inference, we binarize the $\mathbb{M}_{\mathbb{P}}$. In our presented work, we can think of the *RMNs* as replacing the weights of a network with the product of the weights and a binary relevance mixture.

In this work, we introduce two algorithms, **Algorithm 1 and 2 (Supplementary Section B)** which make use of Relevance Mapping. The former is used for traditional *Supervised CL experiments* which used to evaluate *CF* alleviation. The latter is used for the *Unsupervised scenario* (new task detection and unsupervised task inference) concerning evaluation of *CR* alleviation. Importantly, neither of the algorithms relax the conditions of a *strict* CL framework (Section 3.1).[1]

Probabilistic interpretation of Relevance Mapping. French [10] introduced the method of *context-biasing* in which produces internal representations which are both well distributed and well separated to deal with *CF*. *RMN* preserves a similar idea of distribution and separability without constraining for an explicit representation separation amongst posteriors learnt for the sequential tasks. The separation, in turn, is provided by the relevance mappings

$$\mathcal{P}(\theta_1, \mathbb{M}_{\mathbb{P}_1} | D_1) \propto \mathcal{P}(D_1 | \theta_{\mathbb{M}_{\mathbb{P}_1}}) \mathcal{P}(\theta_{\mathbb{M}_{\mathbb{P}_1}}). \tag{6}$$

The $1st$ task of the *CL* problem presented in Eq. (6) is similar to Eq. (1) with relevance mappings introduced under the conditions of the algorithm presented. $\theta_{\mathbb{M}_{\mathbb{P}_i}}$ represents only a subset of θ for which $\mathbb{M}_{\mathbb{P}_i} = 1$. For learning the second task we optimize

$$\mathcal{P}(\theta_{1:2}, \mathbb{M}_{\mathbb{P}_2} | D_{1:2}) \propto \mathcal{P}(D_2 | \theta_{\mathbb{M}_{\mathbb{P}_2}}) \mathcal{P}(\theta_1, \mathbb{M}_{\mathbb{P}_1} | D_1). \tag{7}$$

In Eq.(7), the second term on the right doesn't contribute anything to the optimization over the second task due to the presence of independent relevance mappings which effectively disengages $\theta_{\mathbb{M}_{\mathbb{P}_1}}$ from further tampering and the next task receives a slightly constrained prior distribution that we can refer to as $\theta_2^{''}$. The $\theta_{\mathbb{M}_{\mathbb{P}_1}}$ parameter set is however still available to the second task. For just the 2nd task, Eq.(7) now becomes

$$\mathcal{P}(\theta_{1:2}, \mathbb{M}_{\mathbb{P}_2} | D_2) \propto \mathcal{P}(D_2 | \theta_{\mathbb{M}_{\mathbb{P}_2}}) \mathcal{P}(\theta_2^{''}) \tag{8}$$

which is effectively now a problem of just optimizing an *ANN's* parameters $(\theta, \mathbb{M}_{\mathbb{P}})$ without any dependence on the previous task's posterior. We have effectively decomposed the sequential task parameters. There are three scenarios that may occur w.r.t the optimised parameters i.e. (k represents the individual elements) (i) $\mathbb{M}_{\mathbb{P}_2}^k = \mathbb{M}_{\mathbb{P}_1}^k \Rightarrow \theta_{\mathbb{M}_{\mathbb{P}_1}}^k = \theta_{\mathbb{M}_{\mathbb{P}_2}}^k$ (ii) $\mathbb{M}_{\mathbb{P}_2}^k = 0$ & $\mathbb{M}_{\mathbb{P}_1}^k = 1$ (iii) $\mathbb{M}_{\mathbb{P}_2}^k = 1$ & $\mathbb{M}_{\mathbb{P}_1}^k = 0 \Rightarrow \{\theta_{\mathbb{M}_{\mathbb{P}_1}}^k \cap \theta_{\mathbb{M}_{\mathbb{P}_2}}^k = \emptyset\}$ All of these scenarios can be handled by *RMNs* thanks to the

[1] Refer to Supplementary for further method details

Table 1: Mean test accuracy results for supervised *CL*(Imagenet-50, Split-MNIST, Permuted-MNIST, Sequential Omniglot, Cifar-100 (20 tasks)*(RES-CIFAR)* & Split Cifar-100(10 tasks))

ALGORITHM	IMAGENET-50	P-MNIST	S-MNIST	S-OMNIGLOT	RES-CIFAR	S-CIFAR100
VCL[37]$^{\mathcal{R},\mathcal{H}}$	–	90 *(200 p/t)*	97 *(40 p/t)*	$53.9_{\pm 2.3}$ *(3 p/c)*	–	–
HAT[45]$^{\mathcal{H}}_{\dagger}$	–	91.6	99	$5.5_{\pm 11.1}$	$23.6_{\pm 8.8}$	$59.2_{\pm 0.7}$
RWALK[5]$^{\mathcal{R},\mathcal{H}}$	–	–	82.5	$71.0_{\pm 5.6}$	70.1	$58.1_{\pm 1.7}$
AGS-CL[16]$^{\mathcal{H}}_{\dagger}$	–	–	–	$82.8_{\pm 1.8}$	$27.6_{\pm 3.6}$	$64.1_{\pm 1.7}$
FRCL[50]$^{\mathcal{R}}$	–	$94.3_{\pm 0.2}$ *(200 p/t)*	$97.8_{\pm 0.7}$ *(40 p/t)*	$81.5_{\pm 1.6}$ *(3 p/c)*	–	–
MEGA-II[13]$^{\mathcal{R},**}$	–	91.21 *(256 p/t)*	–	–	$66.1_{\pm 1.9}{}^{\mathcal{M}}$ *(1300 p/t)*	–
DGM-W[38]	17.8	–	–	–	–	–
CGN[1]	35.3	–	–	–	–	–
SNOW([51])$^{\wedge}_{\dagger}$	–	–	–	$82.8_{\pm 1.8}$	–	–
FROMP([40])$^{\mathcal{R}}$		$94.9_{\pm 0.1}$ *(40 p/t)*	$99.0_{\pm 0.1}$ *(40 p/t)*	–	–	–
DLP([48])	–	82	61.2	–	–	–
EWC([20])	–	84	63.1	$67.43_{\pm 4.7}{}^{\mathcal{H}}$	$42.67_{\pm 4.24}{}^{\mathcal{H}}$	$60.2_{\pm 1.1}{}^{\mathcal{H}}$
SI([52])	–	–	57.6	$54.9_{\pm 16.2}$	$45.49_{\pm 0.2}{}^{\mathcal{H}}$	$60.3_{\pm 1.3}{}^{\mathcal{H}}$
MAS[3]$^{\mathcal{H}}_{\dagger}$	–	–	–	$81.4_{\pm 1.8}$	$42_{\pm 1.9}$	$61.5_{\pm 0.9}$
RMN (OURS)	**67.1**$_{\pm 2.1}$	**97.8**$_{\pm 0.1}$	**99.5**$_{\pm 0.2}$	**85.3**$_{\pm 1.7}$	**80.1**$_{\pm 0.9}$	**70.1**$_{\pm 2.5}$

† SIMILAR METHODS(SECTION 2) $\checkmark$ USES PRETRAINED NETWORK $^{\mathcal{R}}$ USES DATA REPLAY BUFFER $^{\wedge}$ ADDITIONAL MODEL IS USED
$^{\mathcal{H}}$ MULTIHEADED LAYER IMPLEMENTATION ** NOT TRAINED OVER ALL TASKS $^{\text{P/T,C}}$ POINTS/TASKS, CHARACTER

Optimal Overlap($\mathcal{O}_2$) hypothesis. For optimizing over n sequential tasks, (8) becomes,

$$\mathcal{P}(\theta_{1:n}, \mathbb{M}_{\mathbb{P}}|D_{1:n}) \propto \prod_{i=1}^{n} \mathcal{P}(D_i|\theta_{\mathbb{M}_{\mathbb{P}_i}})\mathcal{P}(\theta_i^{''}) \tag{9}$$

Looking at (9) which is a basic Bayesian expression for a normal *ANN*, we can now understand that $\mathcal{O}_2$ *hypothesis* inspired *RMN* algorithm is capable of learning well separated and well distributed internal representations thanks to the *posterior decomposition* induced by our method. This takes care of the problem of *CF* and since the parameters of the model are jointly optimized over both the *RMN* parameters θ and the relevance mappings $\mathbb{M}_{\mathbb{P}}$, the network cannot overgeneralize to a specific task given only θ which, in turn takes care of *CR*.

The focus in *RMNs* is not to force a zero representational overlap or just generalize to all the sequential tasks altogether but rather to utilize the *over-parameterization* property of *ANNs* [9] and learn an optimal representational overlap for all tasks in the weight space - corroborating the *Optimal Overlap Hypothesis*. Therefore, there's no constraint on the maps $\mathbb{M}_{\mathbb{P}}$ to minimize the overlap with each other or a global loss function which takes in account of the loss of individual tasks.The map $\mathbb{M}_{\mathbb{P}}$ for each task helps define a subset of the final weight mapping of the *ANNs*. This subset may be disjoint or overlapping with other $\mathbb{M}_{\mathbb{P}}$ defined weight subsets. Since, all the sequential tasks' parameter mappings are subsets of the final weight mapping (with $\mathbb{M}_{\mathbb{P}}$ defining the set relationship), we are able to alleviate both CF (the final mappings generalizes well for all the tasks) and CR (the $\mathbb{M}_{\mathbb{P}}$ preserve the relationship between the global and individual parametric mappings).

5 Experiments

5.1 Catastrophic Forgetting (Supervised Continual Learning)

[2] For *CF* alleviation evaluation, we use the most common CL experimental setup - supervised classification of sequentially available tasks. Even though *RMNs* faithfully follow the *strict CL setup*, we compare it with many current *SOTA* methods which relax the strict constraints.

Setup. We use a wide range of baseline neural architectures (MLPs, CNNs[27], Siamese Networks[22], Residual Networks[14], etc.) to show *RMNs* versatility. As is common in related work

[2]For detailed discussion about hyperparameters, architecture, complexity, etc. please refer to Supplementary

Table 2: *CL* without Task Labels (Mean Test Accuracy)

ALGORITHM	P-MNIST	S-MNIST	S-OMNIGLOT	FUZZY S-MNIST	FUZZY P-MNIST
FRCL$^{\mathcal{R}\triangledown}$	$94.3_{\pm.2}$	$97.8_{\pm.7}$	$81.5_{\pm1.6}$	CND	CND
FROMP$^{\mathcal{R}\triangledown}$	$94.9_{\pm.1}$	$99_{\pm.1}$	–	CND	CND
CN-DPM[28]$^{\mathcal{R},\mathcal{A}\triangledown}$	–	$97.53_{\pm.3}$	–	$93.22_{\pm.1}$	-
RMN (OURS)	$97.7_{\pm.1}$	$99.5_{\pm.2}$	$85.3_{\pm1.7}$	$99.1_{\pm.5}$	$96.9_{\pm1.7}$

^{CND} CAN NOT DO $^{\mathcal{R}}$ USES DATA REPLAY BUFFER $^{\mathcal{A}}$ MULTIPLE NETWORKS USED $^{\triangledown}$ *CR* NOT ALLEVIATED

[50, 20, 37, 40, 16], we evaluate RMNs on six benchmarks: Imagenet-50, Permuted-MNIST[20], Split-MNIST, Sequential Omniglot [44], 10 task Split-Cifar100 [52] and 20 task Split-Cifar100 (Res-Cifar). The Res-Cifar and Imagenet-50 experiments use a Resnet18. For 10 task S-Cifar100, we used 6 convolution layers followed by 2 fully connected layers. MNIST and S-Omniglot architectures are same as in [50]. Refer to Algorithm 1 (Supplementary Section B) for *RMNs*.

For *RMNs*, the classification output $f(x; W, \mathbb{M}_{\mathbb{P}_1}, \ldots \mathbb{M}_{\mathbb{P}_T}) \triangleq argmax_{i \in \{1 \ldots T\}}(\{f(x; W, \mathbb{M}_{\mathbb{P}_\mathbf{i}})\}$ so that no task-specific information is utilized at inference time (showing the inherent task discrimination capability). We also augment the loss function with L1-norm penalty on $\mathbb{M}_{\mathbb{P}}$ masks and sum of overlap of $\mathbb{M}_{\mathbb{P}_{1:T}}$ to reward sparsity and optimal separation of weight spaces, respectively.

Results and Discussion As seen in Table 1, **RMNs set the state of the art across presented *CL* benchmarks** in a strict setup. They also perform robustly against many current SOTA methods which do not follow strict CL setup with improvements of 2.8% (P-mnist), 0.5% (S-mnist), 3.9% (S-Omnliglot), 8.7% (S-Cifar100), 13.9% (Res-Cifar) and 7% (Imagenet-50) over the best results. To the best of our knowledge, it appears that our work is also SOTA when compared to non-strict CL methods in many of the presented benchmarks. Table 1 is divided into two parts with the above part quoting the methods which do not obey the conditions of a strict *CL* framework. Some compared methods[37, 50, 5, 13, 40] employ data replay buffers while others cannot work efficiently without one or more multi-headed layers [37, 5, 16] or a version of it - for e.g. [45] use manual hard coding of layers per task. The lower part of Table 1 consists of methods which are (or can be) implemented in a *strict* sequential learning setup. Unlike most of the compared methods, **RMNs do not require any replay buffers, ensemble networks, meta networks, multi-headed layers or pretrained models** and yet are able to outperform many methods which do. We also compare RMNs with similar methods as mentioned in Section D and see that RMNs substantially outperforms every one of them as seen in Table 1.

5.2 Catastrophic Remembering Experiments

How to evaluate *CR* alleviation? Though more research needs to be done to come up with more robust evaluation metrics, we trivially propose two experiments to evaluate how well a method deals with *CR*. Inspired by the fact that *CR* affects an *ANN's* capability to discriminate between old and new task data and to retain the distinction between all the different tasks that it has trained on, we setup experiments to evaluate the same.

1. *Unsupervised New Task Detection*
2. *Unsupervised Task Inference* - Under this *novel* setup, the model has to identify which task a data input belongs to amongst all the tasks at inference time. From a practical point of view, knowing which task in the sequential task list does the current inference data element belongs to, without human intervention opens up huge opportunities for automation and analysis.

Setup

(i) For new task detection, the model is given no task information during training (and inference) time. The model has to detect a new task in an unsupervised way. The results are then compared with the full supervised version (Table 1) and any performance degradation from the supervised results allow us to evaluate how well the model can alleviate CR. Here, we assume that no information of task labels is given, including the number of disparate tasks.

(ii) After the *ANN* has been trained, the test data is randomized and provided to the model for inference without its task identity (something which would happen in real world *CL* scenario). The

model identifies the task to which the data belongs to and then the test accuracy is calculated from the correctly identified task over the entire task set.
For clarification, since there are no individual heads for each class per task and the training is unsupervised, methods like the ones working under Class Incremental setup[42] do not trivially work.

RMN Methodology (refer to Algorithm 2 (Supplementary Section B))
(i) We initialize f_θ with only a single $\mathbb{M}_\mathbb{P}$, i.e. only a single forward inference path can be learned at initialization, as seen in Line 2 of Algorithm 2 (Supplementary Section B). We set the current task indicator(est_j) to 0. Then, for each minibatch x encountered, we run a task-switch-detection ($TSD(x)$). If $TSD(x)$ returns True, then est_j is incremented and another set of $\mathbb{M}_\mathbb{P}$ is added to f. We use a *Relevance* modified Welsh's t-test on the KL divergence between prior and posterior distributions of the model to determine a task switch [50, 15, 29].

(ii) For *RMNs*, as task j is not given at inference time, thus $max_k f(x, k; W)$ is returned, as seen in Algorithm 2 (Suppl. Sec. 2). Our experimental results show that for any ground-truth task label j, indeed the desired result is $f(x, j; W) \approx max_k f(x, k; W)$, which allows for unsupervised CL inference, as the pathways of different tasks don't overlap unless the tasks are the same.

Results and Discussion
(i) Few methods [50, 28, 40] have tried to deal with the harder problem of learning continually without task labels and none of these follow a strict *CL* framework. [50] and [40] both employ a data replay buffer whereas [28] uses generative replay and a mixture of expert models (which leads to a large increase in computational and memory requirements). Most methods try to sidestep the problem of *CR* in this case by either focusing on the data (instead of model) for task detection or save exemplars to compare incoming data with. Neither actually works on the *CR* problem in the *ANN* - underlining the importance of the strict *CL* setup. However, for comparative purposes, we still display the results of non-strict *CL* methods as well in Table 2. Further, these methods, being heavily dependent on data, fail when the task data is noisy and doesn't come in clean batches. *RMN*s, unlike other methods in Table 2, can easily deal with this since only if *RMNs* have seen the data do they have high and confident activations before calculating the KL divergence test between the prior and posterior to detect the presence of a new set of data. We refer it to as *Fuzzy Unsupervised Learning* setting (with results for *S-MNIST* also in Table 2. Additionally, for comparative purposes only since its the only method (to the best of our knowledge) which can deal with a *fuzzy* setup, Table 2 includes results from[28].But this method is a data-replay, multi ANN model and does not alleviate model *CR*.

(ii) Unfortunately, we couldn't find any *SOTA CL* method which is capable of unsupervised task inference in a *strict CL* setup. In Figure 2a, we show how our algorithm is able to detect the right task - we see that the relevance-weight combination achieves the correct maximum activation in the final layer only when the correct relevance is used. We also display the percentage of correct activations for other relevance values even if they are not maximum activations. Due to lack of comparative methods and space, rest of results for Unsupervised Fuzzy Learning and Unsupervised Task Inference are provided in the Supplementary. According to our knowledge, our method is the only known continual learning method under *strict CL* setup constraints capable of successfully accomplishing unsupervised task inference.

6 Conclusion

In this work, we study the twin problem of catastrophic forgetting and remembering in continual learning. To resolve them, we introduce Relevance Mapping for continual learning, which applies a relevance map on the parameters of a neural network that is learned concurrently to every task. In particular, Relevance Mapping learns an optimal overlap of network parameters between sequentially learned tasks, reducing the representational overlap for dissimilar tasks, while allowing for overlap in the network parameters for related tasks. We demonstrate that our model efficiently deals with catastrophic forgetting and remembering, and **achieves SOTA performance across a wide range of popular benchmarks** without relaxing the conditions of a strict continual learning framework. We do not see any direct potential negative societal impacts of our work. Future work would include making the mapping distribution initialization parameters to be trainable which can be used to calculate the relevance mappings on the fly.

References

[1] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional Channel Gated Networks for Task-Aware Continual Learning. *arXiv:2004.00070 [cs, stat]*, March 2020. arXiv: 2004.00070.

[2] Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. Uncertainty-based Continual Learning with Adaptive Regularization. page 11.

[3] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory Aware Synapses: Learning What (not) to Forget. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, volume 11207, pages 144–161. Springer International Publishing, Cham, 2018. Series Title: Lecture Notes in Computer Science.

[4] Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *J. Mach. Learn. Res.*, 6:1817–1853, December 2005.

[5] Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[6] Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[8] Sayna Ebrahimi, Mohamed Elhoseiny, Trevor Darrell, and Marcus Rohrbach. Uncertainty-guided Continual Learning with Bayesian Neural Networks. April 2020.

[9] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019.

[10] Robert French. Dynamically constraining connectionist networks to produce distributed, orthogonal representations to reduce catastrophic interference, 1994.

[11] Robert M. French. Using semi-distributed representations to overcome catastrophic forgetting in connectionist networks. 1991.

[12] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[13] Yunhui Guo, Mingrui Liu, Tianbao Yang, and Tajana Rosing. Improved schemes for episodic memory-based lifelong learning, 2020.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[15] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks, 2016.

[16] Sangwon Jung, Hongjoon Ahn, Sungmin Cha, and Taesup Moon. Continual learning with node-importance based adaptive group sparse regularization, 2020.

[17] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. 11 2017.

[18] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks, 2017.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[20] 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.

[21] Jeremias Knoblauch, Hisham Husain, and Tom Diethe. Optimal Continual Learning has Perfect Memory and is NP-hard. *arXiv:2006.05188 [cs, stat]*, June 2020. arXiv: 2006.05188.

[22] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

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

[24] Richard Kurle, Botond Cseke, Alexej Klushyn, Patrick van der Smagt, and Stephan Günnemann. Continual Learning with Bayesian Neural Networks for Non-Stationary Data. April 2020.

[25] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[26] Y. LECUN. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*.

[27] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series.

[28] Janghyeon Lee, Hyeong Gwon Hong, Donggyu Joo, and Junmo Kim. Continual Learning With Extended Kronecker-Factored Approximate Curvature. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8998–9007, Seattle, WA, USA, June 2020. IEEE.

[29] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks, 2018.

[30] Stephan Lewandowsky and Shu-Chen Li. 10 - catastrophic interference in neural networks: Causes, solutions, and data. In Frank N. Dempster, Charles J. Brainerd, and Charles J. Brainerd, editors, *Interference and Inhibition in Cognition*, pages 329 – 361. Academic Press, San Diego, 1995.

[31] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting, 2019.

[32] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, Dec 2018.

[33] Sandeep Madireddy, Angel Yanguas-Gil, and Prasanna Balaprakash. Neuromodulated neural architectures with local error signals for memory-constrained online continual learning, 2020.

[34] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In Vittorio Ferrari, Cristian Sminchisescu, Yair Weiss, and Martial Hebert, editors, *Computer Vision – ECCV 2018 - 15th European Conference, 2018, Proceedings*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 72–88. Springer-Verlag Berlin Heidelberg, January 2018. 15th European Conference on Computer Vision, ECCV 2018 ; Conference date: 08-09-2018 Through 14-09-2018.

[35] Martial Mermillod, Aurélia Bugaiska, and Patrick BONIN. The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in Psychology*, 4:504, 2013.

[36] Martin Mundt, Yong Won Hong, Iuliia Pliushch, and Visvanathan Ramesh. A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning, 2020.

[37] Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning, 2017.

[38] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019.

[39] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F. Grewe. Continual learning with hypernetworks. September 2019.

[40] Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard E. Turner, and Mohammad Emtiyaz Khan. Continual deep learning by functional regularisation of memorable past, 2021.

[41] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review, 2018.

[42] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning, 2017.

[43] A. Robins. Catastrophic forgetting in neural networks: the role of rehearsal mechanisms. In *Proceedings 1993 The First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems*, pages 65–68, Dunedin, New Zealand, 1993. IEEE Comput. Soc. Press.

[44] 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. *arXiv preprint arXiv:1805.06370*, 2018.

[45] Serr, Suris, Miron, and Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. *ArXiv*, abs/1801.01423, 2018.

[46] Noel Sharkey and Amanda Sharkey. An analysis of catastrophic interference. *Connect. Sci.*, 7:301–330, 01 1995.

[47] Noel E. Sharkey and Amanda J. C. Sharkey. Backpropagation discrimination geometric analysis interference memory modelling neural nets. *Connection Science*, 7(3-4):301–330, 1995.

[48] Alexander Smola, Vishy Vishwanathan, and Eleazar Eskin. Laplace propagation. 01 2003.

[49] P. K. Srijith and Shirish Shevade. Gaussian process multi-task learning using joint feature selection. In Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 98–113, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[50] Michalis K. Titsias, Jonathan Schwarz, Alexander G. de G. Matthews, Razvan Pascanu, and Yee Whye Teh. Functional Regularisation for Continual Learning with Gaussian Processes. April 2020.

[51] Chungkuk Yoo, Bumsoo Kang, and Minsik Cho. SNOW: Subscribing to Knowledge via Channel Pooling for Transfer & Lifelong Learning of Convolutional Neural Networks. April 2020.

[52] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence, 2017.

[53] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

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] See Section 3, Section 4, Table 1 and Table 2.
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6.
   (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] Please refer to the Supplementary material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Supplementary Section C
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Table 1 and Table 2
   (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)? [No] Since our method (*RMN*) does not have any significant computational or memory overhead over the base methods and since the method is largely network-agnostic, we do not have any new information to discuss. We do however discuss the computational complexity of our work as compared with the base methods.
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 have included citations to the publicly available datasets that we have used for experiments.

(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? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

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]

Supplementary

A Concepts

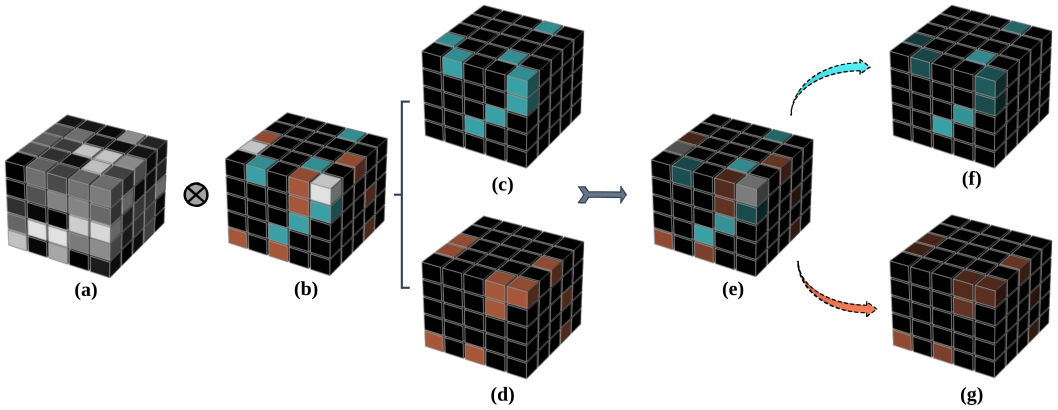

Figure 1: *RMN* example for a single weight matrix - *chromatic colors* represent different tasks, *grayscale* values represent weight node magnitudes and solid *black* nodes (in *(b)* - *(g)*) represent *off* nodes **(a)** a single weight matrix in an *ANN*; **(b)** binary relevance mapping for two tasks - here *white* refers to the node belonging to both tasks and *black* refers to unused nodes; **(c)** & **(d)** show what an individual relevance mapping from *(b)* would look like for each task; **(e)** result of relevance mapping application on weight matrix *(a)*; **(f)** & **(g)** show what a weight matrix would look like for each task during a forward pass.

A.1 Strict Continual Learning

In Section 3 of the main paper, the concept of *Strict Continual Learning* is defined and it is noted that most of the current state of the art Continual Learning methods relax the constraints of a *strict* continually learning framework. In Table 3, we quote some of the major violations of a *strict continual learning* framework with reference to the state of the art methods compared in the main paper.

Data Replay refers to the usage of old or future task data in any way to train the neural network. Methods like [13, 17, 37, 50, 40, 5], etc. all employ this tactic in unique ways to learn continually. It often involves saving old task data in memory modules and been originally inspired from [43] who was among the first researchers to show that data replay helps in alleviating catastrophic forgetting in artificial neural networks, albeit at an expense of relaxing the constraints of a *strict* continual learning setup.

Multihead usually refers to the usage of different last (usually linear) layer for each task. This has become particularly common in continual learning benchmarks with many methods [5] dissuading the usage of single heads for a continually learning neural network. There are a few methods which also employ similar but unique methodology. For e.g. [45] uses a binary hard coded final layer per task.

Pretrained refers to using a pretrained model (usually trained on a more complex dataset like ImageNet [7]) for training on a simpler problem or dataset (e.g. CIFAR [23]). Having used data outside of the continual learning problem setup and since the model has now probably *over-generalized* to entire sequel data/task set, using a pretrained model relaxes the constraints of a *strict* continual learning.

Generative Replay ordinarily refers to the usage of generative modelling for the dataset or task at hand and using these generated samples for some form of data replay. The usage of additional generative model (even for simple classification models and tasks), saving old data points, etc. all violate the conditions of a *strict* continual setup.

Multi-Models refers to the usage of a neural model other than the original model to help with the continual learning problem. This can reflect in using meta networks, or in strategies similar to the one

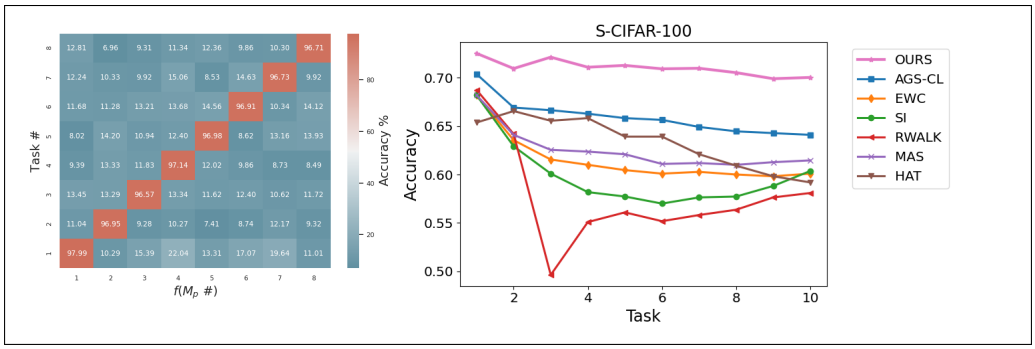

Figure 2: (a) P-MNIST Task Inference (b) Average accuracy results on CIFAR-100 (10 tasks)

presented in [51] where separate *delta* models are used per task to help the original neural network learn continually.

The aforementioned concepts are not mutually exclusive and clearly do not present an exhaustive list of methods employed to relax a *strict* continual learning framework, however they do provide us a reference among-st the compared *SOTA* methods to ascertain which method displays the best results with least amount of constraint relaxation.

Table 3 show that our method, *RMNs*, do not need to use any of the aforementioned methods to violate *strict continual learning* constraints and still produces the state of the art results in common Continual Learning benchmarks.

Table 3: Common Methods used in Continual Learning which relax a *strict CL* framework

Algorithm	Data Replay	Multihead	Pretrained	Generative Replay	Multi-Models
VCL	✓	✓	✗	✗	✗
HAT	✗	✓	✗	✗	✗
RWALK	✓	✓*	✗	✗	✗
AGS-CL	✗	✓	✗*	✗	✗
FRCL	✓	✓	✗	✗	✗
MEGA-II	✓	✓*	✗	✗	✗
SNOW	✗	✓	✓	✗	✓
FROMP	✓	✓*	✗	✗	✗
CGN	✓	✓	✓	✓	✓
DGM-W	✓	✓	✗	✓	✓
DLP	✗	✗	✗	✗	✗
EWC	✗	✓*	✗	✗	✗
SI	✗	✓*	✗	✗	✗
MAS	✗	✓	✗*	✗	✗
RMN (Ours)	✗	✗	✗	✗	✗

* Exceptions exist

Why strict CL? A strict setup is valuable in developing methods for practical cases which involve highly resource constrained *CL* systems e.g. edge devices, remote systems, etc. where changes to the already defined system and saving old task data would be undesirable, economically prohibitive or even impossible. Conceptually, it forces neural systems to deal with *CF* and *CR* on its own with minimal rsources and without reliance on data assumptions or other systems/models. This, in turn, drives research towards earning a better understanding of the properties of neural CL.

Balancing Forgetting and Remembering Having gained a basic understanding about *CF* and *CR*, an astute reader would realize the crux of our problem. Alleviating *CF* appears to aggravate *CR*. While current literature focuses on alleviating CF, the problem of CR does not receive much attention. One aim of this work is to shed light on the twin problem of catastrophic forgetting and remembering and to introduce a method which can balance alleviating both problems concurrently.

A.2 Catastrophic Remembering

A common solution for the problem of *Catastrophic Forgetting* is to *overgeneralize* to the entire set of sequential data/tasks. For example, methods which employ data replay, pre-training and knowledge distillation directly employ *over-generalization* for CF alleviation.

Over-generalization The concept of over-generalization (in the case of back propagated artificial neural networks) refers to the learning of parameter set by the neural network tries to or has already learned a much more general function than what is required.

A simple example to understand *Catastrophic Remembering* was provided in the work by French [12] where a network has a task of reproducing an input as output. A new input is detected if output diverges by a large margin. If the network learns too well and learns the identity function, then it has *overgeneralized* and hence loses the ability to detect new input. This trivial example presents one aspect of *Catastrophic Remembering*. However, there is no guarantee that the loss of discrimination always leads to correct generalization - the network just becomes too familiar with the input irrespective of whether the output is correct.

A.3 Optimal Overlap Hypothesis

Optimal Overlap Hypothesis: *For a strictly continually trained deep neural network, catastrophic forgetting and remembering can be minimized, without additional memory or data, by learning optimal representational overlap, such that the representational overlap is reduced for unrelated tasks and increased for tasks that are similar.*

More formally, for an *ANN* $f_\Theta(D)$ with parameter Θ and sequentially available data D_i over number of tasks/data $i \in [1, \mathbb{T}]$ instead of trying to enforce *over-generalization* which is to learn a superset parameter space which encompasses the sequential tasks $\{\theta_i \mid \cup\theta_{1\ldots\mathbb{T}} = \Theta \wedge \cap\theta_{1\ldots\mathbb{T}} = \emptyset\}$ or complete separation of weight space $\{\theta_i \mid \cup\theta_i \subsetneq \Theta\}$, we try and learn optimal overlaps amongst the sequentially learned parameter sets. That means $\{\forall i, j \in \mathbb{T} \ni i \neq j \mid \theta_i \cap \theta_j = \mathcal{A} \wedge \cup\theta_i = \Theta\}$ where $\mathcal{A} \in [\emptyset, \theta_{i/j}]$.
This hypothesis contrasts from the general concepts of generalization or complete removal of representational overlap. As mentioned earlier, the twin problems *CR* and *CF* can't usually be alleviated at the same time given that generalizing well to all sequential tasks/data leads to aggravating the *CR* in the *ANN*.

B Relevance Mapping Method

Figure 1 shows a simple example of a relevance mapping in effect over a single weight matrix of a artificial neural network.

B.1 Algorithms

Two algorithms using *RMNs* has been cited in the main draft with the first one used primarily in traditional supervised *CL* for classification tasks while the second one is used for the new unsupervised *CL* experiments which are used for *CR* alleviation.

A point to note that adding regularization to induce sparsity in *RMNs* is optional and is not required to obtain an optimally trained model (as shown in Section B.3). Additionally, *model weights are never pruned* in *RMN* methodology. The *pruning* mentioned in Algorithm 1 and 2 refers to zeroing out of relevance mappings which have not tightened towards value of 1. The prune parameter μ may also refer to the combination of weight and $\mathbb{M}_P$.

B.1.1 Catastrophic Forgetting (Supervised)

In the supervised continual learning setup, task labels are available both during training and inference (though *RMNs* do not requires task labels as such). This kind of experimental setup is currently the most common form of evaluation used for Continual Learning methods. (Algorithm 1)

Algorithm 1 *RMN* Supervised Continual Learning

Input: data x, ground truth y for n tasks, prune parameter μ, corresponding task labels i paired with all x
Given: parameters $\mathbf{W}$ & initilaized relevance mappings $\mathbb{M}_{\mathbb{P}}$
for each task i **do**
  $f(x_i; \mathbf{W}, \mathbb{M}_{\mathbf{P_i}}) \Rightarrow \hat{y}_i = \sigma((W \odot \mathbb{M}_{P_i}) \odot x_i)$
  Compute Loss : $L(\hat{y}_i, y_i)$
  Optional: Add Sparsity Loss : $L(\hat{y}_i, y_i) + (\mathbb{M}_{P_i})_{l_0}$
  Backpropagate and optimize
  Prune $\mathbb{M}_{\mathbb{P}} \leq \mu$ only.
  Stabilize (fix) parameters in f where $\mathbb{M}_{\mathbb{P}} = 1$
end for
Inference: For data x and ground-truth task label i:
Output: $f(x, i; W)$

B.1.2 Catastrophic Remembering Experiments

For Unsupervised learning setup (which is used as a measure of Catastrophic Remembering in our work), we introduce two sub tests - new task/data detection and unsupervised/randomized task inference. (Algorithm 2)

In New Task/Data Detection, task label information is unavailable during both training and inference time and the model has to detect the new task in a unsupervised manner.

Algorithm 2 *RMN* Unsupervised Continual Learning

Input: data x, ground truth y, prune parameter μ
Given: parameters $\mathbf{W}$, $\mathbb{M}_{\mathbb{P}_{est_j}}$ with $est_j = 0$, Task Switch Detection Method *TSD*
for each task j **do**
  Filter input x on $f(x; \mathbf{W}, \mathbb{M}_{\mathbb{P}_{\mathbf{0 \ldots j-1}}})$
  $f(x; \mathbf{W}, \mathbb{M}_{\mathbb{P}_{\mathbf{est_j}}}) \Rightarrow \hat{y}$
  Compute Loss : $L(\hat{y}, y)$
  if $TSD(x)$ is True **then**
    est_j++
    Add $\mathbb{M}_{\mathbb{P}_{est_j}}$ to learn-able parameter list
    $f(x; \mathbf{W}, \mathbb{M}_{\mathbb{P}_{\mathbf{est_j}}}) \Rightarrow \hat{y}$
    Re-Compute Loss : $L(\hat{y}, y)$
  end if
  Backpropagate and optimize
  Sample x_g from standard Gaussian distribution with same shape as x
  $f(x_g; \mathbf{W}, \mathbb{M}_{\mathbb{P}_{\mathbf{est_j}}}) \Rightarrow \hat{y}$
  Compute Loss : $||\hat{y} - 0||_2^2$
  Backpropagate and optimize
  Prune $\mathbb{M}_{\mathbb{P}} \leq \mu$ only.
  Stabilize (fix) parameters in f where $\mathbb{M}_{\mathbb{P}} \approx 1$
end for
Inference: For data x:
Output: $max_k f(x, k; W)$

The usual methodology that is followed in an Unsupervised *CL* learning setup involves *boundary detection* between current and new tasks. **RMNs achieve the state of the art for Unsupervised CL Learning** without the usage of data replay buffer, mixture of expert models or any kind of generative replay, unlike [50, 40, 28]. Ordinarily, these task detection methods employ statistical tests like Welch's T-test over clean batches of data (the entire batch data belongs to either the current or the next task). This methodology fails when the incoming previous and the incoming batch are noisy with the incoming batch consisting of the new task data as well as old data. *RMN*s however can easily deal with this by filtering the incoming batch via final layer activations - only if *RMNs* have seen the data do they have high and confident activations before calculating the KL divergence test

between the prior and posterior to detect the presence of a new set of data. However, none of the methods [50, 28] mentioned in the main draft are capable of deploying over such a noisy data setup and are unable to learn continually. [28] does employ a *Fuzzy Testing* scenario for Split-MNIST in which there are transition phases between tasks where the amount of new data increases linearly in each batch. The Fuzzy Unsupervised Learning experiment done on Sequential-MNIST follows the procedure laid out in [28].

The (Randomized) Unsupervised Task Inference experiment also capitalizes on the aforementioned weakness of the former test. Under this, a continually trained model has to identify the task ID during inference under circumstances where the inference input is task randomized. However, since *RMNs* form unique subnetworks in the original neural network, they are capable of trivially identifying the task ID as well as filtering out the noisy training data for new tasks. Under this *novel* setup, the algorithm has to identify at inference time which task a data input belongs to amongst all the tasks it has learned. From a practical point of view, knowing which task in the sequential task list does the current inference data element belongs to, without human intervention opens up huge opportunities for automation and analysis. The average mean test accuracy for Unsupervised Task Inference are: average accuracy: S-Mnist: 99.5 ± 0.2, S-Cifar100: 70 ± 3.1, Res-Cifar: 80 ± 1.2, Imagenet-50: $83.1 \pm .5$

B.2 β parameter

As presented in the main text, we used a sigmoidal pseudo-round function during training which is completely rounded during inference:

$$\mathcal{L}_R(x_k; \beta) = \frac{1}{1 + \exp(-(\beta(x_k - 0.5)))} \tag{10}$$

We also noted that $\lim_{\beta \longrightarrow \infty}(\mathcal{L}_R(x, \beta))$ for $x \in [0, 1]$ is equivalent to the rounding function. Here, β is a learnable, layer-wise parameter (i.e., in our implementation, there is one specific β for every layer of a given network) that controls the "tightness" of $\mathcal{L}_R$. We experimented with different values of β and noted that for an arbitrary high value of β (≥ 80), there's not any visible difference in results and that the β value doesn't require any tuning. If we tried and learn the β parameter instead of fixing its value, we noted that the β value tightened over time.

B.3 Sparsity Analysis

Sparsity($\mathcal{S}$) in a *RMN* $f(\mathcal{W}, \mathbb{M}_P)$ is calculated according to the following formula

$$\mathcal{S} = \frac{\prod_{i=1}^{n} \cap \mathbb{M}_P = 0}{num(\mathcal{W})} \tag{11}$$

Figure 3 shows the model sparsity and usage for Resnet-18 model trained on Cifar-100 [23] dataset over 20 tasks. For this experiment, *there are no loss functions, regularization or any method employed to constrain the model parameters or $\mathbb{M}_P$ for sparsity.* If we do however constrain for sparsity using L_1 or L_0 regularization, we can observe much higher sparsity levels. We can observe that the model capacity usage evens out over time which can be explained due to subsequent tasks finding overlap amongst old task parameters.

B.4 Model Computational Complexity w.r.t number of Tasks

RMNs require only the learned weights of the continually learning network, though this is achieved through creating distinct sub-network mappings in each network. This does increase the number of parameters, but ultimately reduces the effective model size because all additional parameters can be converted to binary tensors. Thus, the memory complexity can be written as $O(tk)$ where t is an integer and equal to the number of tasks and k is a constant. Thus, for a finite and *constrained* value of t, the memory complexity of *RMNs* is $O(1)$ i.e. constant. The value of k depends on the amount of overlap in our model as well as the method used to save binary parameters. For e.g. For the model in Table 5, the theoretical worst case scenario (a model with no overlap amongst the relevance mappings) results in 12kb of memory. Practically, as noticed in the *RMN* sparsity discussion, the model has not been observed to fully utilize its weights over the period of sequential tasks and unused parameters

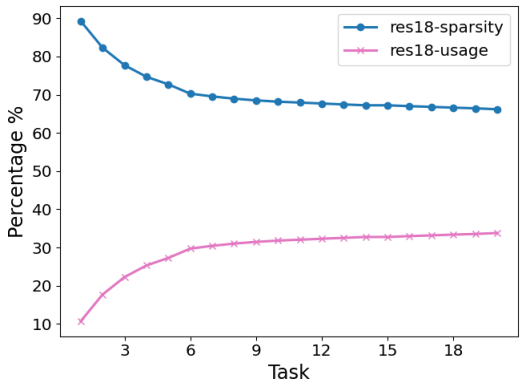

Figure 3: Model Sparsity for AMN-Resnet-18 trained on CIFAR-100 (20 tasks)

Table 4: Additional Mean Accuracy Results for Supervised Experiment

ALGORITHM	IMAGENET-50	CIFAR100(10 TASK)	SPARSITY	SIZE (Mb)
CGN[1]	35.24%	-	-	> 100
L2G[31]	-	79.59% ($resnet26$)	0%	> 60
NNA[33]	-	83.17% ($resnet18 + 50$)	0%	-
RMN (OURS)	$67.1_{\pm 2.1}$ %	$84_{\pm .9}$ ($resnet18$)	52%, 19%	35.3; 45

can be effectively removed post training. Additionally, we do not implement bias parameters in our *RMN*. Thus, effectively, the final model memory footprint is actually *negative* as compared with even the baseline model for all the experiments.

B.5 The Lottery Ticket Hypothesis and Relevance Mapping

A question arises as to whether the slight constraints introduced in the weight space by our algorithm worsen the performance of the sequential tasks. The Lottery ticket Hypothesis introduced in a seminal work [9] states that - *A randomly-initialized, dense neural network contains a subnetwork that is initialized such that—when trained in isolation—it can match the test accuracy of the original network after training for at most the same number of iterations.* Additionally, our method doesn't remove the previous tasks parameters and subsequent methods can choose to use their predecessors parameters optimally. Therefore, no performance drop is expected in our method and results from our experiments prove the same.

C Experimental Details

The experimental implementation for most comparative methods mentioned in Table 3 have been taken from official implementations of [45], [50], [16] and [40].

C.1 Architectural Details

In this section, we provide detailed descriptions of the architectures used for our experiments. We denote 2D convolutional layers as Conv2D, linear layers as Linear, Rectified Linear Unit as ReLU, and Batch Normalization as BN. For *RMN* versions of each layer with included Relevance Mapping $\mathbb{M}_{\mathbb{P}}$, we add an "M-" prefix, e.g. M-Conv is the *RMN* version of Conv.

For fair comparison purposes, for all architectures, we attempt to keep architectural representational capacity and module sequences as similar as possible to referenced methods.

Tablea 5 and 6 provide the main architecture details of the continual learning experiments dealing with Sequential MNIST, Permuted MNIST and Sequential Omniglot benchmarks.

Table 5: Details of the Multi-layer Perceptron network used for the Permuted-MNIST and Split-MNIST tasks. This architecture is of equivalent representational capacity to the network used in [50].

INPUT: $x \in \mathbb{R}^{784}$
M-LINEAR $(784) \longrightarrow 100$
M-BN (100)
RELU
M-LINEAR $(100) \longrightarrow 100$
M-BN (100)
RELU
M-LINEAR $(100) \longrightarrow 100$
M-BN (100)
RELU
M-LINEAR $(100) \longrightarrow 10$
SOFTMAX

For the sequential Cifar-100 (10 tasks) [23] benchmark, we follow the experimental and architectural details from [16]. For the *RMN*, however, we do not use bias parameters and dropout layers. Table 7 shows the architecture details for the *RMN* used in the experiment.

For the *RES-CIFAR* experiment which uses a Resnet-18[14][3] trained over Sequential Cifar-100 (20 tasks), we use the original Resnet-18 architecture for all experiments with modifications as required by a specific method. For *Imagenet-50* experiment, we used a Resnet-18 as well. For the *RMN*-Resnet-18, we do not make use of bias parameters and dropout layers.

C.2 Hyperparameter Details

In this section, we provide detailed descriptions of training, optimization, and hyperparameter details(Table 8).

In [50] and [16], they select experiment with a range of hyperparameters, choose the values that return the highest validation accuracy and then retrain on the union of the train and validation set. When applicable, we select hyperparameter values similar or equivalent to those arrived at in [50] for MNIST and Omnliglot experiments and [16] for Cifar-100 experiments. For all continual learning tasks, we make use of the Adam optimizer and have separate learning rates for weights and $\mathbb{M}_{\mathbb{P}}$ parameters. Subsets of weights are frozen via gradient masking as tasks increase, where $\prod_t^T \mathbb{M}_{\mathbb{P}_\approx} = 1$ is the mask applied to the weights at task $T+1$.

For the Permuted-MNIST and Split-MNIST tasks, we use a 90-10 train-test split, 0.002 learning rate for all parameters, and batch size of 128. For all tasks, the network is trained for 250 epochs.

For the Sequential Omniglot task, we use an 80-20 train-test split, 0.0002 learning rate for all parameters, except $\mathbb{M}_{\mathbb{P}}$ parameters, a learning rate of 0.0001 for $\mathbb{M}_{\mathbb{P}}$ parameters, a batch size of 16. For the first task, the network is trained for 150 epochs, for subsequent tasks the network trained for 200 epochs.

In S-CIFAR-100 and RES-CIFAR, we train all comparative methods with mini-batch size of 256 for 100 epochs using Adam optimizer[19] with initial learning rate 0.001 and decaying it by a factor of 3 if there is no improvement in the validation loss for 5 consecutive epochs, similarly as in [16, 45].

For Imagenet-50 experiment, we use use a Resnet-18 for our method and use a Resnet-18 or bigger model for other reported methods. For [1] and [38], we report their Imagenet-50 experiments as is reported in their works. Please, refer to their works for further details.

[3] HAT [45] authors mentioned that there is no official implementation for Residual Networks

Table 6: Details of the convolutional network used for the Sequential Omniglot task. This architecture is of similar or equivalent representational capacity to the network used in [50], cited as "Baseline" in their results sections.

INPUT: $x \in \mathbb{R}^{1\times105\times105}$ RESIZE $\longrightarrow x \in \mathbb{R}^{1\times28\times28}$
M-CONV2D $1ch \longrightarrow 250ch$
M-BN (250)
RELU
MAXPOOLING (2 X 2), $stride = 2$
M-CONV2D $250ch \longrightarrow 250ch$
M-BN (250)
RELU
MAXPOOLING (2 X 2), $stride = 2$
M-CONV2D $250ch \longrightarrow 250ch$
M-BN (250)
RELU
MAXPOOLING (2 X 2), $stride = 2$
M-CONV2D $250ch \longrightarrow 250ch$
M-BN (250)
RELU
MAXPOOLING (2 X 2), $stride = 2$
M-LINEAR $\longrightarrow 60$
SOFTMAX

For our method *(RMNs)*, we keep the same mini batch size, training epochs and optimizer as mentioned in [16]. For Split Cifar-100 (10 tasks) and RES-CIFAR (Split Cifar100-20 tasks with Resnet-18), the model weight parameters initial learning rate is .001 and for $\mathbb{M}_P$ the learning rate is .01. The prune parameter value is .05 and .01 respectively which is used to prune the relevance mappings. The pruning is done whenever the model's task loss converges which varies from epoch $20 - 80$ for different tasks.

For compared methods, We ran experiments for *mnist* and *cifar100* experiments with hyperparameters and details as reported in [50] and [16] respectively.

D Similar Methods

The idea of using soft-masking in networks (usually on non-linear activations) has been utilized before in novel ways for solving different problems. However, few of them, if any, ground these methods in some underlying concept (Optimal Overlap in our case) and often these methods include masks which are mutually exclusive, for e.g., in sparsity learning [53], joint learning where Mallya *et al.* [34] *piggyback* a pretrained network by using a non-differentiable mask thresholding function and value, etc. In contrast, we don't require our models to be pretrained. In *CL*, the following methods appear to be closest to our Relevance Mapping Networks (RMNs):

HAT[45] proposes a task based hard attention mechanism which may be considered the most similar to our *RMN*. It differs from *RMN* due to following reasons- (i) *HAT* utilizes task embeddings and a positive scaling parameter - and a gated product of these two is used to produce a non-binary mask - unlike our *RMNs* which don't use either a task embedding or a scaling parameter and is necessarily binary. (ii) Unlike *RMNs*, the attention on the last layer in *HAT* is manually hard-coded for every task. (iii) A recursive cumulative attention mechanism is employed to deal with multiple non binary

Table 7: Details of the convolutional network used for the Sequential Cifar 100 (10 tasks) task. This architecture is of similar or equivalent representational capacity to the network used in [16]

INPUT: $x \in \mathbb{R}^{3\times32\times32}$
M-CONV2D $3ch \longrightarrow 32ch$
RELU
M-CONV2D $32ch \longrightarrow 32ch$
RELU
MAXPOOLING (2 X 2), $stride = 2$
M-CONV2D $32ch \longrightarrow 64ch$
RELU
M-CONV2D $64ch \longrightarrow 64ch$
RELU
MAXPOOLING (2 X 2), $stride = 2$
M-CONV2D $64ch \longrightarrow 128ch$
RELU
M-CONV2D $128ch \longrightarrow 128ch$
RELU
MAXPOOLING (2 X 2), $stride = 2$
M-LINEAR $\longrightarrow 256$
M-LINEAR $\longrightarrow 10$

Table 8: Hyperparameters for *RMNs*

	IMAGENET-50	S-CIFAR100	RES-CIFAR	P-MNIST
BATCH SIZE	256	256	256	128
LEARNING RATE	.001	.001	.001	.001
MAPPING LR (μ)	.01	.01	.01	.01
OPTIMIZER	ADAM	ADAM	ADAM	ADAM
TASKS	5	10	20	10
EPOCHS PER TASK	$130(1^{st}), 100$	$120(1^{st}), 80$	$130(1^{st}), 80$	60
WEIGHT DECAY	0	0	0	0
WEIGHT PRUNE PARAMETER(ω_w)	$.05, .01$	.01	.01	$.22, .2, .1, .01$
MAPPING PRUNE PARAMETER (ω_m)	.999999	.999999	.999999	.999999
ARCHITECTURE	RESNET-18	MLP	RESNET-18	MLP

mask values over tasks in *HAT*. *RMNs* however have no need for such a mechanism. (iv) *HAT* cannot be used in a unsupervised *CL* setup or to deal with *CR* and has not been implemented with more complex network architectures like Residual Networks.

PGD[16] or *proximal gradient descent algorithm* progressively freeze nodes in an *ANN*. (i) Unlike *RMNs*, this method employs selective regularization to signify node importance (which is calculated by lasso regularization). (ii) This method progressively uses up the parameter set of the *ANN* and it is unclear whether it can be used for an arbitrary large number of sequential tasks. (iii) This method is also unable to deal with unsupervised learning scenario or *CR*. (iv) This method uses a different classification layer for each task - relaxing the core constraints of the problem altogether.

MAS[3] (i) calculates the parameter importance by calculating sensitivity via the squared $l2$ norm of the function output to their changes and then uses regularization (similar to [20]) to enforce it, unlike *RMNs*. (ii) *MAS* enforces fixed synaptic importance between tasks irrespective of their similarity and unlike *RMNs*, can't trivially work under unsupervised learning scenarios.

SNOW[51] (i) uses a unique channel pooling scheme to evaluate the channel relevance for each specific task which differs from *RMN's* individual node relevance mapping strategy. (ii) Importantly, *SNOW*, unlike *RMNs*, employs a pre-trained source model which is frozen and already *overgeneralizes* to the *CL* problem at hand and thus makes this method inapplicable for dealing with *CR*. (iii) It also doesn't seem to be capable of handling unsupervised learning/testing scenarios.

D.1 Differences with Relevance Mapping Method

[45] proposes hard attention (*HAT*), a task based attention mechanism which can be considered the most similar to our *RMN*.

It differs from *RMN* due to following reasons-

1. They utilize task embeddings and a positive scaling parameter - and a gated product of these two is used to produce a non-binary mask - unlike our *RMNs* which don't use either a task embedding or a scaling parameter and is necessarily binary.
2. Unlike *RMNs*, the attention on the last layer in *HAT* is manually hard-coded for every task.
3. A recursive cumulative attention mechanism is employed to deal with multiple non binary mask values over tasks in *HAT*. *RMNs* however have no need for such a mechanism.
4. *HAT* cannot be used in a unsupervised *CL* setup or to deal with *CR* and has not been implemented with more complex network architectures like Residual Networks.

[16] uses *proximal gradient descent algorithm* to progressively freeze nodes in an *ANN*.

1. Unlike *RMNs*, this method employs selective regularization to signify node importance (which is calculated by lasso regularization).
2. This method progressively uses up the parameter set of the *ANN* and it is unclear whether it can be used for an arbitrary large number of sequential tasks.
3. This method employs two group sparsity-based penalties in order to regularize important nodes, however *AMN* do not require usage such kind of sparse based penalty.
4. This method is also unable to deal with unsupervised learning scenario or *CR*. (iv) This method uses a different classification layer for each task - relaxing the core constraints of the problem altogether.

[3]

1. Calculates the parameter importance by calculating sensitivity of the squared $l2$ norm of the function output to their changes and then uses regularization (similar to [20] to enforce in sequential learning, unlike *RMNs*.
2. The method enforces fixed synaptic importance between tasks irrespective to their similarity and unlike our work, doesn't seem to be capable of working under Unsupervised Learning scenarios.

[51] propose *SNOW* and

1. Uses a unique channel pooling scheme to evaluate the channel relevance for each specific task which differs from *RMN's* individual node relevance mapping strategy.
2. Importantly, this work, unlike *RMNs*, employs a pre-trained model which is frozen source model which already *overgeneralizes* to the *CL* problem at hand and thus makes this method inapplicable for dealing with *CR*.
3. It also doesn't seem to be capable of handling unsupervised learning/testing scenarios.

E Drawbacks and Future Work

At present, the relevance mappings are initialized randomly using a clipped Normal probability density function. This allows most values which are near the mean (0.5) of the initializing normal distribution to learn to tighten towards 0 and 1 accordingly, however values which have been initialized near 0 and 1 require large gradients to allow them to shift to the other end of the spectrum. A more informative initialization methodology would allow for more optimal overlap amongst the task parameters.