Random Path Selection for Incremental Learning

Jathushan Rajasegaran **Munawar Hayat** **Salman Khan**

Fahad Shahbaz Khan **Ling Shao**

Inception Institute of Artificial Intelligence
first.last@inceptioniai.org

Abstract

Incremental life-long learning is a main challenge towards the long-standing goal of Artificial General Intelligence. In real-life settings, learning tasks arrive in a sequence and machine learning models must continually learn to increment already acquired knowledge. Existing incremental learning approaches, fall well below the state-of-the-art cumulative models that use all training classes at once. In this paper, we propose a random path selection algorithm, called RPS-Net, that progressively chooses optimal paths for the new tasks while encouraging parameter sharing. Since the reuse of previous paths enables forward knowledge transfer, our approach requires a considerably lower computational overhead. As an added novelty, the proposed model integrates knowledge distillation and retrospection along with the path selection strategy to overcome catastrophic forgetting. In order to maintain an equilibrium between previous and newly acquired knowledge, we propose a simple controller to dynamically balance the model plasticity. Through extensive experiments, we demonstrate that the proposed method surpasses the state-of-the-art performance on incremental learning and by utilizing parallel computation this method can run in constant time with nearly the same efficiency as a conventional deep convolutional neural network.

1 Introduction

The ability to incrementally learn novel tasks and acquire new knowledge is necessary for life-long machine learning. Deep neural networks suffer from '*catastrophic forgetting*' [18], a phenomenon that occurs when a network is sequentially trained on a series of tasks and the learning acquired on new tasks interferes with the previously learned concepts. As an example, in a typical transfer learning scenario, when a model pre-trained on a source task is adapted to another task by fine-tuning its weights, its performance significantly degrades on the source task whose weights are overridden by the newly learned parameters [13]. It is, therefore, necessary to develop continual learning models capable of incrementally adding newly available classes without the need to retrain models from scratch using all previous class-sets (a cumulative setting). .

An ideal incremental learning model must meet the following criterion. **(a)** As a model is trained on new tasks, it is desirable to maintain its performance on the old ones, thus avoiding catastrophic forgetting. **(b)** The knowledge acquired on old tasks should help in accelerating the learning on new tasks (a.k.a forward transfer) and vice versa. **(c)** As the class-incremental learning progresses, the network must share and reuse the previously tuned parameters to realize a bounded computational complexity and memory footprint, **(d)** At all learning phases, the model must maintain a tight

Codes available at https://github.com/brjathu/RPSnet

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

equilibrium between the existing knowledge base and newly presented information (stability-plasticity dilemma).

Despite several attempts, existing incremental learning models partially address the above mentioned requirements. For example, [16] employs a distillation loss to preserve knowledge across multiple tasks but requires prior knowledge about the task corresponding to a test sample during inference. An incremental classifier and representation learning approach [21] jointly uses distillation and prototype rehearsal but retrains the complete network for new tasks, thus compromising model stability. The progressive network [22] lacks scalability as it grows paths linearly (and parameters quadratically) with the number of tasks. The elastic weight consolidation scheme [15] computes synaptic importance offline using Fisher information metric thus restricting its scalability and while it works well for permutation tasks, its performance suffers on class-incremental learning [12].

Here, we argue that the most important characteristic of a true incremental learner is to maintain the right trade-off between '*stability*' (leading to *intransigence*) and '*plasticity*' (resulting in *forgetting*). We achieve this requisite via a dynamic path selection approach, called RPS-Net, that proceeds with random candidate paths and discovers the optimal one for a given task. Once a task is learned, we fix the parameters associated with it, that can only be shared by future tasks. To complement the previously learned representations, we propose a stacked residual design that focuses on learning the supplementary features suitable for new tasks. Besides, our learning scheme leverages exemplar-based retrospection and introduces an explicit controller module to maintain the equilibrium between stability and plasticity for all tasks. During training, our approach always operates with a constant parameter budget that at max equals to a conventional linear model (e.g., `resent` [6]). Furthermore, it can be straightforwardly parallelized during both train and test stages. With these novelties, our approach obtains state-of-the-art class-incremental learning results, surpassing the previous best model [21] by 7.38% and 10.64% on CIFAR-100 and ImageNet datasets, respectively.

Our main contributions are:

- A random path selection approach that provides faster convergence through path sharing and reuse.
- The residual learning framework that incrementally learns residual paths which allows network reuse and accelerate the learning process resulting in faster training.
- Ours is a hybrid approach that combines the respective strengths of knowledge distillation (via regularization), retrospection (via exemplar replay) and dynamic architecture selection methodologies to deliver a strong incremental learning performance.
- A novel controller that guides the plasticity of the network to maintain an equilibrium between the previously learned knowledge and the newly presented tasks.

2 Related Work

The catastrophic interference problem was first noted to hinder the learning of connectionist networks by [18]. This highlights the stability-plasticity dilemma in neural networks [1] i.e., a rigid and stable model will not be able to learn new concepts while an easily adaptable model is susceptible to forget old concepts due to major parameter changes. The existing continual learning schemes can be divided into a broad set of three categories: **(a)** regularization schemes, **(b)** memory based retrospection and replay, and **(c)** dynamic sub-network training and expansion.

A major trend in continual learning research has been on proposing novel regularization schemes to avoid catastrophic forgetting by controlling the plasticity of network weights. [16] proposed a knowledge distillation loss [7] which forces the network to retain its predictions on the old tasks. Kirkpatrick et al. [15] proposed an elastic weight consolidation mechanism that quantifies the relevance of parameters to a particular task and correspondingly adjusts the learning rate. In a similar spirit, [28] designed intelligent synapses which measure their relevance to a particular task and consequently adjust plasticity during learning to minimize interference with old tasks.

Rebuffi et al. [21] proposed a distillation scheme intertwined with exemplar-based retrospection to retain the previously learned concepts. [8] considered a similar approach for cross-dataset continual learning [16]. The combination of episodic (short-term) and semantic (long-term) memory was studied in [11, 5, 10] to perform memory consolidation and retrieval. Particularly, [10, 11] help avoid explicitly storing exemplars in the memory, rather using a generative process to recall memories.

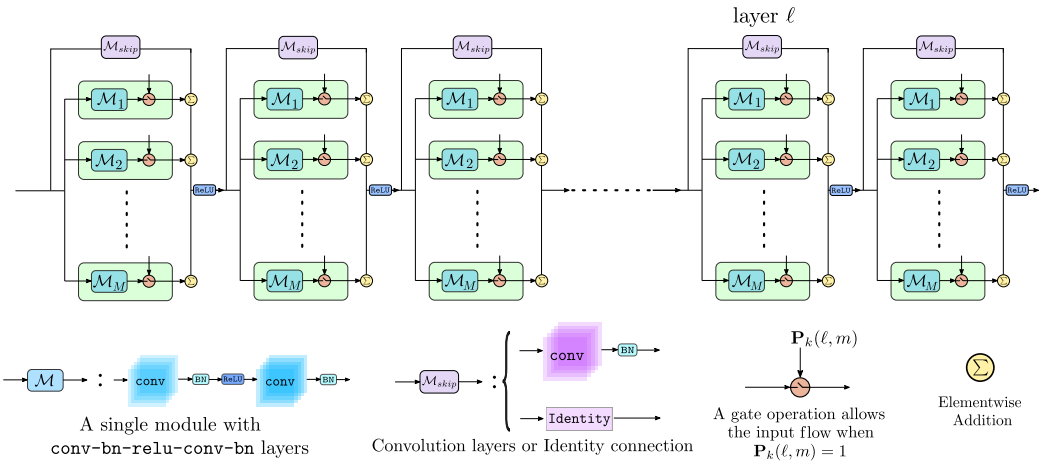


Figure 1: *An overview of our RPS-Net:* The network architecture utilizes a parallel residual design where the optimal path is selected among a set of randomly sampled candidate paths for new tasks. The residual design allows forward knowledge transfer and faster convergence for later tasks. The random path selection approach is trained with a hybrid objective function that ensures the right trade-off between network stability and plasticity, thus avoiding catastrophic forgetting.

The third stream of works explores dynamically adapting network architectures to cope with the growing learning tasks. [22] proposed a network architecture that progressively adds new branches for novel tasks that are laterally connected to the fixed existing branches. Similarly, [26] proposed a network that not only grows incrementally but also expands hierarchically. Specific paths through the network were selected for each learning task using a genetic algorithm in PathNet [4]. Afterwards, task-relevant paths were fixed and reused for new tasks to speed-up the learning efficiency.

The existing adaptive network architectures come with their respective limitations e.g., [22]'s complexity grows linearly with the tasks, [26] has an expensive training procedure and a somewhat rigid architecture and [4] does not allow incrementally learning new classes due to a detached output layer and a relatively expensive genetic learning algorithm used in [4]. In comparison, we propose a random path selection methodology that provides a significant boost and enables faster convergence. Furthermore, our approach combines the respective strengths of the above two types of methods by introducing a distillation procedure alongside an exemplar-based memory replay to avoid catastrophic forgetting.

3 Method

We consider the recognition problem in an incremental setting where new tasks are sequentially added. Assuming a total of K tasks, each comprising of U classes. Our goal is to sequentially learn a deep neural network, that not only performs well on the new tasks but also retains its performance on the old tasks. To address this problem, we propose a random path selection approach (RPS-Net) for new tasks that progressively builds on the previously acquired knowledge to facilitate faster convergence and better performance. In the following, we explain our network architecture, the path selection strategy, a hybrid objective function and the training procedure for incremental learning.

3.1 RPS-Net Architecture

Our network consists of L distinct layers (see Figure 1). Each layer $\ell \in [1, L]$ is constitutes a set of basic building blocks, called modules $\mathcal{M}^\ell$. For simplicity, we consider each layer to contain an equal number of M modules, stacked in parallel, i.e., $\mathcal{M}^\ell = \{\mathcal{M}_m^\ell\}_{m=1}^M$, along with a skip connection module $\mathcal{M}_{skip}^\ell$ that carries the bypass signal. The skip connection module $\mathcal{M}_{skip}^\ell$ is an identity function when the feature dimensions do not change and a learnable module when the dimensions vary between consecutive layers. A module $\mathcal{M}_m^\ell$ is a learnable sub-network that maps the input features to the outputs. In our case, we consider a simple combination of `(conv-bn-relu-conv-bn)` layers for each module, similar to a single `resnet` block [6]. In contrast to a residual block which

consists of a single identity connection and a residual branch, we have one skip connection and M residual blocks stacked in parallel. The intuition behind developing such a parallel architecture is to ensure multiple tasks can be continually learned without causing catastrophic interference with other paths, while simultaneously providing parallelism to ensure efficiency.

Towards the end of each layer in RPS-Net, all the residual connections, as well as skip connections, are combined together using element-wise addition to aggregate complimentary task-specific features obtained from different paths. Remarkably, for the base-case when $M = 1$, the network is identical to a conventional `resnet` model. After the Global Average Pooling (GAP) layer that collapses the input feature maps to generate a final feature $\mathbf{f} \in \mathbb{R}^D$, we use a fully connected layer classifier with weights $\mathbf{W}_{fc} \in \mathbb{R}^{D \times C}$ (C being the total number of classes) that is shared among all tasks.

For a given RPS-Net with M modules and L layers, we can define a path $\mathbf{P}_k \in \mathbb{R}^{L \times M}$ for a task k:

$$\mathbf{P}_k(\ell, m) = \begin{cases} 1, & \text{if the module } \mathcal{M}_m^\ell \text{ is added to the path,} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The path $\mathbf{P}_k$ is basically arranged as a stack of one-hot encoded row vectors $\boldsymbol{e}^{(i)}$ (i-th standard basis):

$$\mathbf{P}_k = \Big\{ \mathbf{P}_k(\ell) \in \{0,1\}^M : \mathbf{P}_k(\ell) = \boldsymbol{e}^{(i)} \equiv \sum_{m=1}^{M} \mathbf{P}_k(\ell, m) = 1 \Big\}, s.t., i \sim \mathbb{U}\big(\{\mathbb{Z} \cap [1, M]\}\big), \tag{2}$$

where i is the selected module index, uniformly sampled using $\mathbb{U}(\cdot)$ over the set of integers $[1, M]$.

We define two set of paths $\mathbf{P}_k^{tr}$ and $\mathbf{P}_k^{ts}$ that denote the train and inference paths, respectively. Both are formulated as binary matrices: $\mathbf{P}_k^{tr,ts} \in \{0,1\}^{L \times M}$. When training the network, any m^{th} module in l^{th} layer with $\mathbf{P}_k^{tr}(l, m) = 1$ is activated and all such modules together constitute a training path $\mathbf{P}_k^{tr}$ for task k. As we will elaborate in Sec. 3.2, the inference path is evolved during training by sequentially adding newly discovered training paths and ends up in a "*common*" inference path for all inputs, therefore our RPS-Net does not require knowledge about the task an input belongs to. Some previous methods (e.g., [16]) need such information, which limits their applicability to real-world incremental class-learning settings where one does not know in advance the corresponding task for an input sample. Similarly, only the modules with $\mathbf{P}_k^{ts}(\ell, m) = 1$ are used in the inference stage.

3.2 Path Selection

With a total of K tasks, we assume a constant number of U classes that are observed in each k^{th} task, such that $U = C/K$. Without loss of generality, the proposed path selection strategy can also be applied to a variable number of classes occurring in each task. The path selection scheme enables incremental and bounded resource allocation, with progressive learning that ensures knowledge exchange between the old and new tasks resulting in positive forward and backward transfer.

To promote resource reuse during training that in turn improves training speed and minimizes computational requirements, we propose to perform path selection after every J task, where $1<J<K$. As a result, the path selection is performed only $\lceil K/J \rceil$ times in total during the complete training process. Our experiments show that J can be set to a higher value without sacrificing the incremental learning performance (see Sec. 4.3). For every J tasks, N paths are randomly chosen and followed by training process. The best path is then selected from these group of N sub-models and is shared among the next J tasks. Further, we also stop the training of the old modules (i.e., fix their paths and parameters) after the training for a particular group of tasks is completed. Hence, at any point, only L layers with a maximum of one module are being trained.

The random path selection strategy is illustrated in Fig. 2. Our choice of random path generation as a mechanism to select an optimal path is mainly inspired by the recent works of [27, 30, 20]. These works show that random search for an optimal network architecture performs almost the same as other computationally demanding approaches e.g., genetic algorithms and reinforcement learning (RL) based methods. Besides, some incremental learning approaches resort to adding new resources to the network, resulting in network expansion [22, 26]. In contrast, our path selection algorithm does not result in linear expansion of resources since a new path is created only after J tasks and overlapping modules are reused when the new path is intersecting old paths. Further, even when all the modules are exhausted (saturated), the skip connections are always trained. We show via an

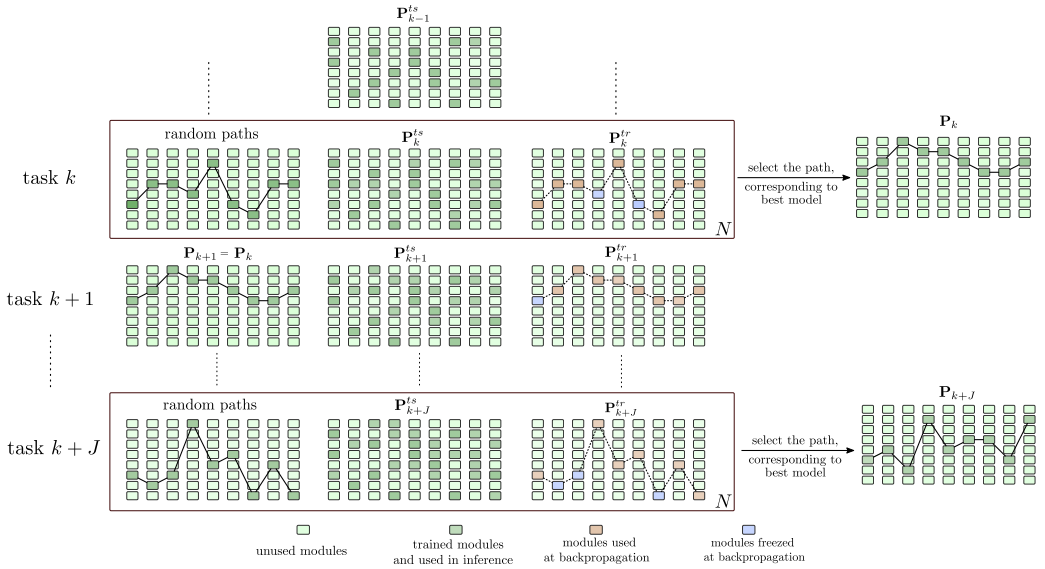


Figure 2: *Path Selection Approach:* Given a task k, N random paths are initialized. For each path, only the modules different from the previous inference path $\mathbf{P}_{k-1}^{ts}$ are used to form the training path $\mathbf{P}_k^{tr}$. Among N such paths, the optimal $\mathbf{P}_k$ is selected and combined with the $\mathbf{P}_{k-1}^{ts}$ to obtain $\mathbf{P}_k^{ts}$. Notably, the path selection is only performed after J tasks. During training, the complexity remains bounded by a standard single path network and the resources are shared between tasks.

extensive ablation study that even when all paths are saturated, our RPS-Net can still learn useful representations as the skip connections and classification layer remains tunable in every case.

At any point in time, we train a single path (equivalent to a `resnet`) while rest of the inference paths are fixed. Due to this, the path we use for a task k essentially learns the residual signal relative to the fixed paths that were previously trained for old tasks. For example, if we are training $\mathbf{P}_k^{tr}$, the weights of $\mathbf{P}_{\lfloor k/J \rfloor}^{ts} \veebar \mathbf{P}_k^{tr}$ are fixed, where $\veebar$ denotes the exclusive disjunction (logical XOR operation). Essentially, the complete $\mathbf{P}_k^{tr}$ is not used for training rather its disjoint portion that has not already been trained for any of the old tasks is learned i.e., $\mathbf{P}_k^{tr} \veebar (\mathbf{P}_k^{tr} \wedge \mathbf{P}_{\lfloor k/J \rfloor}^{ts})$. In this way, previous knowledge is shared across the network via overlapping paths and skip connections. When the network is already trained for several tasks, a new path for the current task only needs to learn higher order residuals of the network. This has an added advantage that convergence becomes faster as we learn more tasks since each new task will be learned taking advantage of the previous information.

The optimal path based on the performance of N path configurations is selected as $\mathbf{P}_k$. All such task-specific paths are progressively combined together to evolve a common inference path $\mathbf{P}_k^{ts}$,

$$\mathbf{P}_k^{ts} = \mathbf{P}_1^{tr} \vee \mathbf{P}_2^{tr} \ldots \vee \mathbf{P}_k^{tr}, \tag{3}$$

where $\vee$ denotes the inclusive disjunction (logical OR) operation. At each task k, the inference path $\mathbf{P}_k^{ts}$ is used to evaluate all previous classes.

3.3 Incremental Learning Objective

Loss function: We use a hybrid loss function that combines regular cross-entropy loss as well as a distillation loss to incrementally train the network.

For a task $k \in [1, K]$ with each task having U classes, we calculate the cross-entropy loss as follows,

$$\mathcal{L}_{ce} = -\frac{1}{n} \sum_i \mathbf{t}_i[1 : k * U] \log(\text{softmax}(\mathbf{q}_i[1 : k * U])), \tag{4}$$

where i denotes the example index, $\mathbf{t}(x)$ is the one-hot encoded true label, $\mathbf{q}(x)$ are the logits obtained from the network's last layer and n is the mini batch size. To keep the network robust to catastrophic

forgetting, we also use distillation loss in the objective function,

$$\mathcal{L}_{dist} = \frac{1}{n}\sum_{i} \text{KL}\Bigg(\log\left(\sigma\left(\frac{\mathbf{q}_i[1:(k-1)*U]}{t_e}\right)\right), \sigma\left(\frac{\mathbf{q'}_i[1:(k-1)*U]}{t_e}\right)\Bigg). \tag{5}$$

Here, σ is the softmax function and t_e is the temperature used in [7] and $\mathbf{q}'(x)$ are the logits obtained from the networks' previous state.

Controller: It is crucial to maintain a balance between the previously acquired learning and the knowledge available from the newly presented task. If the learning is biased towards either of the two objectives, it will result in either catastrophic forgetting (losing old task learning) or interference (obstructing learning for the new task). Since our network is trained with a combined objective function with $\mathcal{L}_{ce}$ and $\mathcal{L}_{dist}$, it is necessary to adequately control the plasticity of the network. We propose the following controller that seeks to maintain an equilibrium between $\mathcal{L}_{ce}$ and $\mathcal{L}_{dist}$,

$$\mathcal{L} = \mathcal{L}_{ce} + \phi(k,\gamma)\cdot\mathcal{L}_{dist}, \tag{6}$$

where, $\phi(k,\gamma)$ is a scalar coefficient function with γ as a scaling factor, introduced to increase the distillation contribution to the total loss. Intuitively, as we progress through training, $\phi(k,\gamma)$ will also increase to ensure that network remembers old information,

$$\phi(k,\gamma) = \begin{cases} 1, & \text{if } k \leq J \\ (k-J)*\gamma, & \text{otherwise.} \end{cases} \tag{7}$$

4 Experiments and Results

4.1 Implementation Details

Dataset and Protocol: For our experiments, we use evaluation protocols similar to iCARL [21]. We incrementally learn 100 classes on CIFAR-100 in groups of 10, 20 and 50 at a time. For ImageNet, we use the same subset as [21] comprising of 100 classes and incrementally learn them in groups of 10. After training on a new group of classes, we evaluate the trained model on test samples of all seen classes (including current and previous tasks). Following iCARL [21], we restrict exemplar memory budget to 2k samples for CIFAR-100 and ImageNet datasets. Note that unlike iCARL, we randomly select our exemplars and do not employ any herding and exemplar selection mechanism.

We also experiment our model with MNIST and SVHN datasets. For this, we resize all images to 32×32 and keep a random exemplar set of 4.4k, as in [9]. We group 2 consecutive classes into one task and incrementally learn five tasks. For evaluation, we report the average over all classes (A_5).

Training: For the CIFAR100 dataset, we use `resnet-18` along with max pooling after 5th, 7th blocks and global average pooling (GAP) after 9th block. For ImageNet dataset, we use the standard `resnet-18` architecture as in [21]. After the GAP layer, a single fully connected layer with weights $\mathbf{W}_{fc} \in \mathbb{R}^{512\times100}$ is used as a classifier. For MNIST, a simple 2 layered MLP (with 400 neurons each), whereas for SVHN `resnet-18` is used, similar to [9].

For each task, we train our model for 100 epochs using Adam [14] with $t_e = 2$, with learning rate starting from 10^{-3} and divided by 2 after every 20 epochs. We set the controller's scaling factor to $\gamma = 2.5$ and $\gamma = 10$ respectively for CIFAR and ImageNet datasets. We use the ratio between the number of training samples for a task and the fixed number of exemplars as the value for γ. We fix $M = 8$ and $J = 2$ except for the 50 classes per task, where $J = 1$. We do not use any weight or network regularization scheme such as dropout in our model. For augmentation, training images are randomly cropped, flipped and rotated ($< 10^0$). For each task, we train $N = 8$ models in parallel using a NVIDIA-DGX-1 machine. These models come from the randomly sampled paths in our approach and may have some parts frozen due to an overlap with previous tasks. Our codes are available https://github.com/brjathu/RPSnet.

4.2 Results and Comparisons

We extensively compare the proposed technique with existing state-of-the-art methods for incremental learning. These include Elastic Weight Consolidation (EWC) [15], Riemannian Walk (RWalk) [3], Learning without Forgetting (LwF) [16], Synaptic Intelligence (SI) [28], Memory Aware Synapses

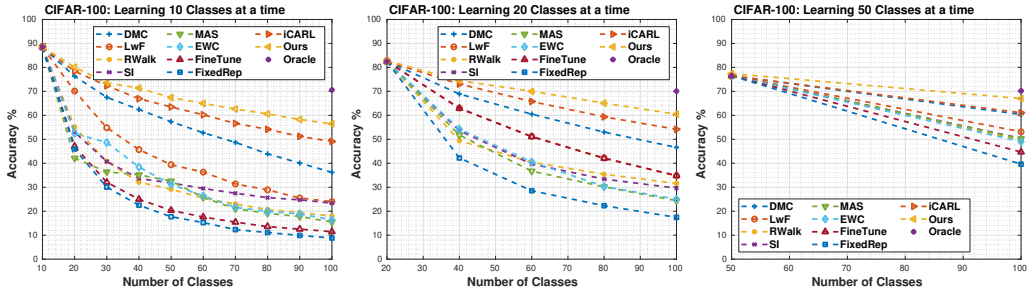


Figure 3: Results on CIFAR-100 with 10, 5 and 2 tasks (from *left* to *right*). We surpass STOA results.

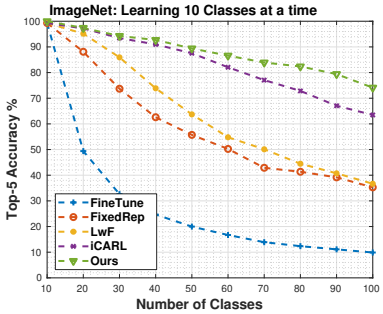


Figure 4: Results on ImageNet dataset for learning 10 classes at a time. We surpass STOA results by 10.3%.

Methods	MNIST(A_5)	SVHN(A_5)
Joint training	97.53%	93.23%
EWC [15]	19.80%	18.21%
online-EWC [23]	19.77%	18.50%
SI [28]	19.67%	17.33%
MAS [2]	19.52%	17.32%
LwF [16]	24.17%	-
GEM* [17]	92.20%	75.61%
DGR* [24]	91.24%	-
RtF* [25]	92.56%	-
RPS-Net*	**96.16%**	**88.91%**

Table 1: Comparison on MNIST and SVHN datasets. Ours is a memory based approach (denoted by '$*$'), and outperforms state-of-the-art.

(MAS) [2], Deep Model Consolidation (DMC) [29] and Incremental Classifier and Representation Learning (iCARL) [21]. We further evaluate on three baseline approaches: Fixed Representation (*FixedRep*) where the convolution part of the model is frozen and only the classifier is trained for newly added classes, *FineTune* where the complete previously learnt model is tuned for the new data, and *Oracle* where the model is trained on all samples from previous and current tasks.

Fig. 3 compares different methods on CIFAR-100 datasets, where we incrementally learn groups of 10, 20 and 50 classes at a time. The results indicate superior performance of the proposed method in all settings. For the case of learning 10 classes at a time, we outperform iCARL [21] by an absolute margin of 7.3%. Compared with the second best method, our approach achieves a relative gain of 5.3% and 9.7% respectively for the case of incrementally learning 20 and 50 classes on CIFAR-100 dataset. For the case of 50 classes per task, our performance is only 3.2% below the *Oracle* approach, where all current and previous class samples are used for training. Fig. 4 compares different methods on ImageNet dataset. The results show that for experimental settings consistent with iCARL [21], our proposed method achieves a significant absolute performance gain of 10.3% compared with the existing state-of-the-art [21]. Our experimental results indicate that commonly used technique of fine-tuning a model on new classes is clearly an inferior approach, and results in catastrophic forgetting. Table 1 compares different methods on MNIST and SVHN datasets following experimental setting of [9]. The results show that RPS-Net, surpasses all previous methods with a margin of 4.3% and 13.3% respectively for MNIST and SVHN datasets. The results further indicate that the methods which do not use a memory perform relatively lower.

4.3 Ablation Studies and Analysis

Contribution from Each Component of RPS-Net: Fig. 5a studies the impact of progressively integrating individual components of our RPS-Net. We begin with a simple baseline model with a single path that achieves 37.97% classification accuracy on CIFAR100 dataset. When distillation loss is used alongside the baseline model, the performance increases to 44.93%. The addition of our proposed controller $\phi(k, \gamma)$ in the loss function further gives a significant boost of $+6.83\%$, resulting in an overall accuracy of 51.76%. Finally, the proposed multi-path selection algorithm along with above mentioned components increases the classification accuracy up to 58.48%. This demonstrates

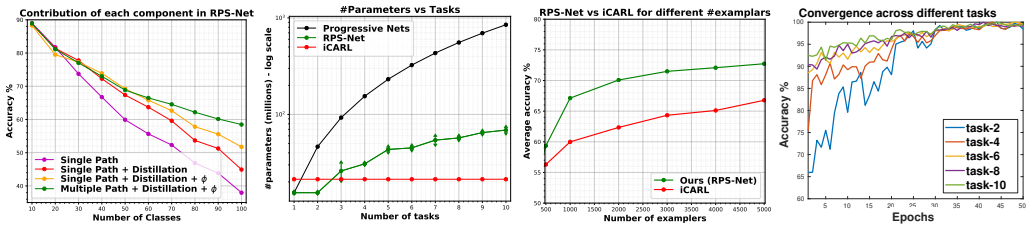
that our two contributions, controller and multi-path selection, provide a combined gain of 13.6% over baseline + distillation.

Increase in the #Parameters: Fig. 5b compares total parameters across tasks for Progressive Nets [22], iCARL [21] and our RPS-Net on CIFAR100. Our model effectively reuses previous parameters, and the model size does not increase significantly with tasks. After 10 tasks, RPS-Net has 72.26M parameters on average, compared with iCARL (21.3M) and Progressive Nets (932.84M). In RPS-Net the number of parameters and FLOPs increase logarithmically, while for Progressive Nets they increase quadratically.

Scaling Factor γ: It controls the equilibrium between cross-entropy and distillation losses (or the balance between new and old tasks). In Fig. 6, for smaller γ, the network tends to forget old information while learning the new tasks well and vice versa. For example, when $\gamma = 1$ (same as loss function used in iCaRL [21]) the performance drops after 5 tasks, showing the model is not at its equilibrium state. On the other hand, $\gamma = 8$ achieves the best performance at earlier task ($2, 3, 4$ and 5), with drop in performance towards the later tasks (51% at task 10). Empirically, we found the optimal value for $\gamma = 2.5$, to keep the equilibrium till last tasks.

Varying Blocks and Paths: One of the important restriction in RPS-Net design is the networks' capacity, upper-bounded by $M \times L$ modules. As proposed in the learning strategy, a module is trained only once for a path. Hence, it is interesting to study whether the network saturates for a high number of tasks. To analyze this effect, we change the parameter M and J. Our results with varying M are reported in Fig. 6, which demonstrate that the network can perform well even when all paths are saturated. This effect is a consequence of our residual design where skip connections and last classification layer are always trained, thus helping to continually learn new tasks even if the network is saturated. If saturation occurs, the model has already learned the generalization of input distribution, hence, a residual signal (carrying complementary information) via skip connections is enough to adjust to a new task. Further, once the network has seen many tasks, it learns generalizable features that can work well for future tasks with adaptation of the final classification layer weights.

In Fig. 6, we illustrate results with varying paths (paths $\propto \frac{1}{J}$) in the network. We note that learning a high number of paths degrades performance as the previously learned parameters are less likely to be effectively reused. On the other hand, we obtain comparable performance with fewer paths (e.g., 2 for CIFAR-100).



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

Figure 5: From *left to right*: (a) Contribution from each component of the RPS-Net, (b) Increase in number of parameters with number of tasks, (c) RPS-Net performance on different memory sizes and (d) Forward transfer showing faster convergence as the tasks increase.

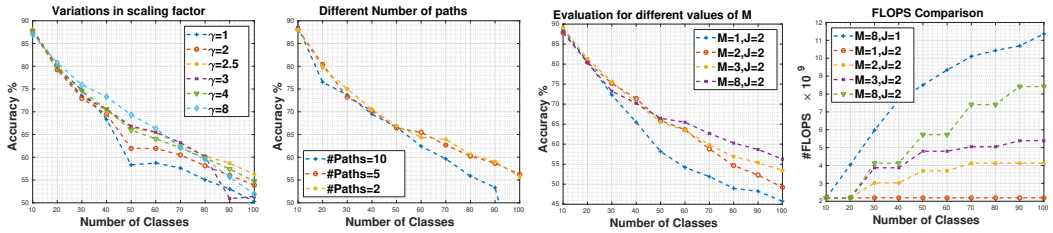


Figure 6: From *left to right*: Ablation analysis for parameters γ, J & M and the number of FLOPS.

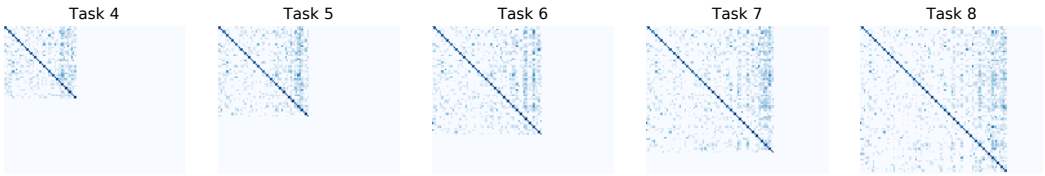


Figure 7: Confusion matrices over 10 incremental tasks on CIFAR-100, showing backward knowledge transfer.

Difference from Genetic Algorithms: We compare our random selection with a genetic algorithm i.e., Binary Tournament Selection (BTS) for 25 maximum generations, on MNIST with 5 tasks (each of 2 classes), using a simple 2 layer (100 neurons) MLP with M = 8, J = 1. On 5 runs, our proposed random selection achieves an average accuracy of 96.52% vs BTS gets 96.32%. For same time complexity as ours, BTS has an average accuracy of 71.24% for the first generation models. For BTS to gain similar performance as our random selection, it needs an average of 10.2 generations (> # random paths), hence BTS has more compute complexity. Sophisticated genetic algorithms may beat random selection with a small margin, but likely with a high compute cost, which is not suitable for an incremental classifier learning setting having multiple tasks.

Forward Transfer: The convergence trends shown in Fig. 5d demonstrate the forward knowledge transfer for RPS-Net. We can see that for task-2, the model takes relatively longer to converge compared with task-10. Precisely, for the final task, the model achieves 95% of the total performance within only one epoch, while for the second task it starts with 65% and takes up-to 20 epochs to achieve 95% of the final accuracy. This trends shows the faster convergence of our model for newer tasks This effect is due to residual learning as well as overlapping module sharing in RPS-Net design, demonstrating its forward transfer capability.

Backward Transfer: Fig. 7 shows evolution of our model with new tasks. We can see that the performance of the current task (k) is lower than the previous tasks ($<k$). Yet, as the model evolves, the performance of task k gradually increases. This demonstrates models' capability of backward knowledge transfer, which is also reflected in biological aspects of human brain. Specifically, hippocampus in human brain accomplishes fast learning which is later slowly consolidated with the slow learning at neocortex [19]. In Fig. 7, we can see the pattern of slow learning, with the performance on new tasks gradually maturing. We also quantitatively validate Backwards Transfer with BWT metric (see Eq. 3 in GEM [17], larger the better). After last task, BWT values are -0.1462 (RPS-Net) vs. -0.4602 (iCARL) which shows the better backward transfer capability of our model.

FLOPS comparison: As the number of tasks increase, the network's complexity grows. As shown in Fig. 6, with different configurations of modules and paths, the computational complexity of our approach scales logarithmically. This proves that the complexity of RPS-Net is bounded by $\mathcal{O}(log(\#task))$. This is due to the fact that the overlapping modules increase as the training progresses. Further, in our setting we chose new paths after every $J > 1$ tasks. Hence, in practice our computational complexity is well below the worst-case logarithmic curve. For example with a setting of M=2, J=2 the computational requirements reduces by 63.7% while achieving the best performance. We also show that even when a single path is used for all the tasks (M=1), our model achieves almost the same performance as state-of-the-art with constant computational complexity.

5 Conclusion

Learning tasks appear in a sequential order in real-world problems and a learning agent must continually increment its existing knowledge. Deep neural networks excel in the cumulative learning setting where all tasks are available at once, but their performance deteriorates rapidly for incremental learning. In this paper, we propose a scalable approach to class-incremental learning that aims to keep the right balance between previously acquired knowledge and the newly presented tasks. We achieve this using an optimal path selection approach that support parallelism and knowledge exchange between old and new tasks. Further, a controlling mechanism is introduced to maintain an equilibrium between the stability and plasticity of the learned model. Our approach delivers strong performance gains on MNIST, SVHN, CIFAR-100 and ImageNet datasets for incremental learning problem.

References

[1] W. C. Abraham and A. Robins. Memory retention–the synaptic stability versus plasticity dilemma. *Trends in neurosciences*, 28(2):73–78, 2005.

[2] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154, 2018.

[3] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547, 2018.

[4] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

[5] A. Gepperth and C. Karaoguz. A bio-inspired incremental learning architecture for applied perceptual problems. *Cognitive Computation*, 8(5):924–934, 2016.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[8] S. Hou, X. Pan, C. Change Loy, Z. Wang, and D. Lin. Lifelong learning via progressive distillation and retrospection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 437–452, 2018.

[9] Y.-C. Hsu, Y.-C. Liu, A. Ramasamy, and Z. Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. 2018.

[10] N. Kamra, U. Gupta, and Y. Liu. Deep generative dual memory network for continual learning. *arXiv preprint arXiv:1710.10368*, 2017.

[11] R. Kemker and C. Kanan. Fearnet: Brain-inspired model for incremental learning. *International Conference on Learning Representations*, 2018.

[12] R. Kemker, M. McClure, A. Abitino, T. L. Hayes, and C. Kanan. Measuring catastrophic forgetting in neural networks. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[13] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun. A guide to convolutional neural networks for computer vision. *Synthesis Lectures on Computer Vision*, 8(1):1–207, 2018.

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. In *Proceedings of the national academy of sciences*, volume 114, pages 3521–3526. National Acad Sciences, 2017.

[16] Z. Li and D. Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2018.

[17] D. Lopez-Paz et al. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017.

[18] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[19] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *CoRR*, abs/1802.07569, 2018.

[20] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

[21] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[22] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[23] J. Schwarz, J. Luketina, W. M. Czarnecki, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell. Progress & compress: A scalable framework for continual learning. *arXiv preprint arXiv:1805.06370*, 2018.

[24] H. Shin, J. K. Lee, J. Kim, and J. Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017.

[25] G. M. van de Ven and A. S. Tolias. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*, 2018.

[26] T. Xiao, J. Zhang, K. Yang, Y. Peng, and Z. Zhang. Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 177–186. ACM, 2014.

[27] S. Xie, A. Kirillov, R. Girshick, and K. He. Exploring randomly wired neural networks for image recognition. *arXiv preprint arXiv:1904.01569*, 2019.

[28] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3987–3995. JMLR. org, 2017.

[29] J. Zhang, J. Zhang, S. Ghosh, D. Li, S. Tasci, L. Heck, H. Zhang, and C.-C. J. Kuo. Class-incremental learning via deep model consolidation. *arXiv preprint arXiv:1903.07864*, 2019.

[30] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.