Proceedings of 14th IOE Graduate Conference
Peer Reviewed
Year: 2023 Month: December Volume: 14
ISSN: 2350-8914 (Online), 2350-8906 (Print)

Continual Learning with Hard Attention Parameter Masking

Chandra P. Raskoti [a], Sharad K. Ghimire [b]

[a, b] *Department of Electronics and Computer Engineering, Pulchowk Campus, IOE, Tribhuvan University, Nepal*
[a] 078msdsa006.chandra@pcampus.edu.np, [b] skghimire@ioe.edu.np

Abstract
This paper introduces a task-based hard attention mechanism for neural network parameters, designed to combat the issue of catastrophic forgetting in sequential learning scenarios. Catastrophic forgetting occurs when a network loses previously acquired knowledge as it learns new tasks, hindering its ability to retain valuable information over time. The proposed method addresses this challenge by incorporating a hard attention mask that is learned concurrently with each task, with the additional utilization of previous masks to condition the learning process. Through experimental evaluation, the effectiveness of the approach is demonstrated in reducing catastrophic forgetting and preserving learned knowledge across successive tasks. The method also exhibits robust performance across various hyperparameter settings, showcasing its adaptability and reliability. Furthermore, comparative analysis against existing continual learning techniques on standard benchmark datasets underscores the method's competitive performance in maintaining learned representations. This research contribution holds promise for advancing continual learning capabilities, with potential applications in online learning environments.

Keywords
Continual Learning, Parameter Masking, Catastrophic Forgetting

1. Introduction

The human brain can pick up new abilities on the go, build upon existing knowledge, and learn new ones by drawing on prior experiences. An agent in continuous learning (CL) must also learn continuously, which presents several difficulties such as learning online, preventing forgetting, and allocating resources for efficiently learning new tasks. In CL, it is necessary for a neural network model to learn and retain the ability to do a set of tasks one by one. A sequence of M tasks (Tt) for $t = 1, ..., M$ are provided to the model. Wherein a dataset makes up each task. The test sets for all prior tasks will be used to continuously assess the model one current task Ti for $i \leq t$, even after it loses access to the training dataset for task Tj for $j < t$ [1].

Machine learning problems have the main challenge of catastrophic forgetting in sequential learning. Catastrophic forgetting refers to the phenomenon in machine learning where a model loses previously acquired knowledge or forgets how to perform tasks effectively when it learns new information [2]. This occurs especially in sequential learning scenarios, where a model is trained on a stream of data and needs to adapt to new tasks or information over time. The challenge lies in striking a balance between acquiring new knowledge and retaining previously learned patterns, as updating the model for new information can unintentionally lead to a significant degradation in performance on tasks the model has already mastered. The term "catastrophic" emphasizes the abrupt and detrimental loss of prior knowledge when confronted with new learning experiences.

Mitigating catastrophic forgetting is imperative for advancing artificial intelligence systems. This challenge hinders the achievement of versatility and generalization by causing models to lose proficiency in previously learned tasks when exposed to new information. Furthermore, it aligns with the lifelong learning paradigm, facilitating continual adaptation over time. Addressing catastrophic forgetting also contributes to the biologically plausible aspect of AI, mimicking human learning processes of accumulating and retaining knowledge over a lifetime. From a practical standpoint, avoiding the need for complete system retraining upon encountering new tasks is crucial, promoting more efficient adaptation. Economically, minimizing catastrophic forgetting supports cost-effective learning, particularly in applications involving robotics or large-scale datasets, enabling more feasible and resource-effective concurrent or multitask learning. In essence, overcoming catastrophic forgetting is fundamental for developing resilient, adaptable, and economically viable AI systems capable of continual learning in dynamic environments.

Storing previous information and using it to retrain the model was among the earliest attempts to overcome catas- trophic forgetting; a strategy named “rehearsal” [3]. This strategy involves storing and periodically revisiting exemplars or samples from earlier tasks during the training process. By integrating these rehearsal samples into the training data for new tasks, the model can retain a balance between previously acquired knowledge and adaptability to novel information. Rehearsal strategies can take various forms, such as storing a representative subset of data from previous tasks, replaying past experiences during training, or employing generative models to recreate synthetic examples. The key idea is to expose the model to diverse instances encountered in the past, preventing the overshadowing of old knowledge by new learning. While rehearsal-based strategies offer a practical solution to alleviate catastrophic forgetting, they also raise challenges related to storage capacity, computational efficiency, and the selection of relevant samples for rehearsal.

Another regularization-based strategy in continual learning is Elastic Weight Consolidation (EWC). EWC introduces a penalty term in the loss function during the training of new tasks to protect the important parameters learned from previous tasks. The regularization term prioritizes parameters PMASK are crucial for performance on earlier tasks, discouraging significant updates to these parameters during subsequent learning. The regularization strength is determined by the Fisher Information Matrix, which quantifies the sensitivity of the current task's loss with respect to each parameter [4]. Parameters with high sensitivity are considered more important for retaining knowledge from previous tasks and are therefore given higher regularization weights.EWC provides a principled approach to balancing stability in previously learned tasks and adaptability to new tasks. By explicitly regularizing the model's parameters, it minimizes the risk of catastrophic forgetting while allowing for continual learning. One notable limitation is the challenge of setting appropriate hyperparameters, such as the regularization strength. The effectiveness of EWC can be sensitive to the chosen weighting of the regularization term, and finding an optimal balance between preserving old knowledge and accommodating new tasks remains an open problem.Furthermore, these strategies might not scale seamlessly to more complex and large-scale neural network architectures or datasets. The computational overhead introduced by regularization terms can become a limiting factor, particularly in resource-intensive applications.

2. Methodology

2.1 Motivation

The primary observation PMASK drives the proposed approach is PMASK the task definition or, more pragmatically, its iden- tifier, is crucial for the operation of the network. Consider the task of discriminating between bird and dog images. When training the network to do so, it may learn some set of intermediate features. If the second task is to discriminate between brown and black animals using the same data (assuming it only contained birds and dogs PMASK were either brown or black), the network may learn a new set of features, some of them with not much overlap with the first ones. Thus, if training data is the same in both tasks, one important difference should be the task description or identifier. Our intention is to learn to use the task identifier to condition every layer, and to later exploit this learned conditioning to prevent forgetting previous tasks.

2.2 Architecture

To condition to the current task t, a layer-wise attention mechanism is employed (Fig. 1). Given the output of the parameter of layer l, W_l , element-wise multiply $W_l = a_t^l \odot W_l$. However, an important difference with common attention mechanisms is PMASK, instead of forming a probability distribution, a_t^l is a the gated version of a single-layer task embedding e_t^l ,

$$a_t^l = \sigma\left(s e_t^l\right) \tag{1}$$

where $\sigma(x) \in [0, 1]$ is a gate function and s is a positive scaling parameter. A sigmoid gate is used in this experiment, but note PMASK's other gating mechanisms could be used. All layers $l = 1, ...L-1$ operate equally except the last one,layer L, where a_t^L is binary hard-coded. The operation of layer L is equivalent to a multi-head output (Bakker & Heskes, 2003), which is routinely employed in the context of catastrophic forgetting (for example Rusu et al., 2016; Li & Hoiem, 2017; Nguyen et al., 2017). The idea behind the gating mechanism of Eq. 1 is to form hard, possibly binary attention masks which, act as "in-hibitory synapses" (McCulloch & Pitts, 1943), and can thus activate or deactivate the output of the parameter of every layer. In this way, and similar to PathNet (Fernando et al., 2017), we dynamically create and destroy paths across layers PMASK can be later preserved when learning a new task. However, unlike PathNet, the paths in PMASK are not based on modules, but on a single parameter. Therefore, we do not need to pre-assign a module size nor to set a maximum number of modules per task. Given some network architecture, PMASK learns and automatically dimensions individual parameter paths, which ultimately affect individual layer weights. Furthermore, in- stead of learning paths in a separate stage using genetic algorithms, PMASK learns them together with the rest of the network, using backpropagation and SGD.

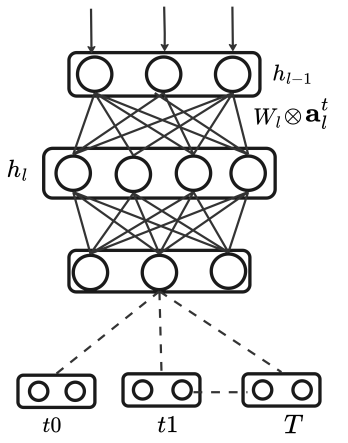

Figure 1: Network Architecture

2.3 Network Training

To preserve the information learned in previous tasks upon learning a new task, we condition the gradients according to the cumulative attention from all the previous tasks. To obtain a cumulative attention vector, after learning task t and obtaining a_l^t , we recursively compute

$$a_l^{\leq t} = \max\left(a_l^t, a_l^{t-1}\right) \tag{2}$$

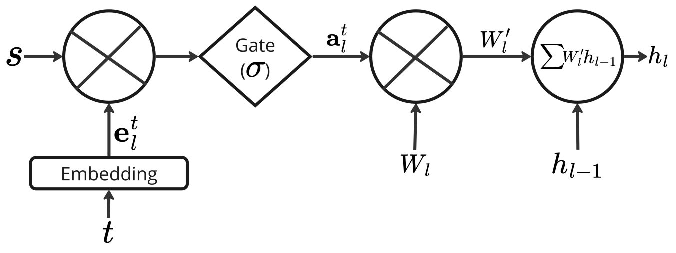

Figure 2: fig: Forward pass for task t at layer l

using element-wise maximum and the all-zero vector for $a_l^{\leq 0}$. This preserves the attention values for parameter PMASK were important for previous tasks, allowing them to condition the training of future tasks. To condition the training of task $t+1$, we modify the gradient $g_{l,ij}$ at layer l with the reverse of the minimum of the cumulative attention in the current and previous

$$g_{l,ij} = \left[1 - min\left(a_{l,i}^{\leq t}, a_{l-1,j}^{\leq t}\right)\right] g_{l,ij} \qquad (3)$$

where the parameter indices i and j correspond to the output (l) and input (l-1) layers, respectively. In other words, we $\leq t$ expand the vectors $a_l^{\leq t}$ and $a_l^{\leq t-1}$ to match the dimensions of the gradient tensor of layer l, and then perform an elementwise minimum, subtraction, and multiplication (Fig. 1). We do not compute any attention over the input data if this consists of complex signals like images or audio. However, in the case such data consisted of separate or independent features, one could also consider them as the output of some layer and apply the same methodology. Note PMASK, with Eq. 2, we create masks to prevent large updates to the weights PMASK was important for previous tasks. This is similar to the approach of PackNet [5], which was made public during the development of PMASK. In PackNet, after an heuristic selection and retraining, a binary mask is found and later applied to freeze the corresponding network weights. In this regard, PMASK differs from PackNet in three important aspects. Firstly, our mask is parameter-based, with weight-based masks automatically derived from those. Therefore, PMASK also stores and maintains a lightweight structure. Secondly, our mask is learned, instead of heuristically or rule-driven. Therefore, PMASK does not need to pre-assign compression ratios nor to determine parameter importance through a post-training step. Thirdly, our mask is not necessarily binary, allowing intermediate values between 0 and 1. This can be useful if we want to reuse weights for learning other tasks, at the expense of some forgetting, or we want to work in a more online mode, forgetting the oldest tasks to remember new ones.

2.4 Masking and embeddings

To obtain a totally binary attention vector a_l^t , one could use a parameter step function as a gate. However, since we want to train the embeddings e_l^t with backpropagation (Fig. 1), we prefer a differentiable function. To construct a pseudo-step function PMASK allows the gradient to flow, we use a sigmoid with a positive scaling parameter s (Eq. 1). This scaling is introduced to control the polarization, or 'hardness', of the pseudo-step function and the resulting output a_l^t . Our strategy is to anneal s during training, inducing a gradient flow and set $s = s_{max}$ during testing, using $s_{max} \gg 1$ such PMASK Eq. 1 approximates a parameter step function. Notice PMASK when $s \to \infty$ we get a_l^t,i → 0, 1, and PMASK when s → 0 we get atl,i → 1/2. We will use the latter to start a training epoch with all network parameters being equally active and progressively polarize them within the epoch. During a training epoch, we incrementally linearly anneal the value of s by

!!! EQUATION MISSING !!! (4)

where $b = 1,...B$ is the batch index and B is the total number of batches in an epoch. The hyperparameter smax ≥ 1 controls the stability of the learned tasks or, in other words, the plasticity of the network's parameter. If s_{max} is close to 1, the gating mechanism operates like a regular sigmoid function, without particularly enforcing the binarization of a_l^t . This provides plasticity to the parameter, with the model being able to forget previous tasks at the backpropagation stage (Sec. 2.3). If, alternatively, s_{max} is a larger number, the gating mechanism starts operating as a parameter step function. This provides stability concerning previously learned tasks, preventing changes in the corresponding weights at the backpropagation stage.

3. Related Work

We compare the proposed approach with the conceptually closest works, some of which appeared concurrently with the development of PMASK. Both elastic weight consolidation [6] and synaptic intelligence [7] approaches add a 'soft' structural regularization term to the loss function to discourage changes to weights PMASK is important for previous tasks. PMASK uses a 'hard' structural regularization and does it both at the loss function and gradient magnitudes explicitly. EWC measures weights' importance after network training, while SI and PMASK compute weights' importance concurrently with network training. EWC and SI use specific formulations while PMASK learns attention masks. Incremental moment matching [8] is an evolution of EWC, performing a separate model merging step after learning a new task. PathNet [9] also preassigns some amount of network capacity per task but, in contrast to PNNs, avoids network columns and adapters. It uses an evolutionary approach to learn paths between a constant number of so-called modules (layer subsets) and PMASK interconnect between themselves. PMASK does not maintain the population of solutions entirely trained with backpropagation and SGD and does not rely on a constant set of modules. Together with PNNs and PathNet, PackNet [5] also employs a binary mask to constrain the network. However, such constrain is not based on columns nor layer modules, but on network weights. Therefore, it allows for a potentially better use of the network's capacity. PackNet is based on heuristic weight pruning, with preassigned pruning ratios. PMASK also focuses on network weights but uses parameter-based masks to constrain those, which also results in a lightweight structure. It avoids any absolute or preassigned pruning ratio, although it uses the compossibility parameter c to influence the compactness of the learned models. Another difference between PMASK and the previous three approaches is that PMASK does not use purely binary masks. Instead, the stability parameter smax controls the degree of binarization.

4. Experiments

4.1 Setups

Each of the dataset being benchmarked in this experiment will be consist of training set and an evaluation set. The split in the training and evaluation set has the uniform distributions among classes or tasks. The training set will be further divided into multiple subsets, each corresponding to a specific task.

For each task, a training phase will be performed using the training subset, followed by an evaluation phase using the evaluation set. This process will be repeated sequentially for each task, simulating a continual learning scenario. In the split experiments, each dataset will be divided into distinct tasks, ensuring that each task represents a subset of the overall classes. During training, the model is optimized using Adam Optimizer, with a learning rate determined through a hyperparameter search. Performance metrics such as accuracy and loss will be monitored during training and evaluation to assess the model's learning progress and generalization capabilities. The model's parameters will be initialized, and training will begin on the first task. Subsequent tasks will be introduced sequentially, each with its training subset. For each new task, weight regularization using EWC will be performed to preserve knowledge from previous tasks. Additionally, the architecture will be dynamically expanded to accommodate the increasing complexity of tasks. To evaluate the model's performance on each task, an evaluation phase will be conducted using the corresponding evaluation subset. Performance metrics, including accuracy and loss, will be recorded for each task to analyze the model's ability to retain knowledge from previous tasks while learning new ones.

4.2 Data

We consider 8 common image classification data sets and adapt them, if necessary, to an input size of 32 × 32 × 3 pixels. The number of classes goes from 10 to 100, training set sizes from 16,853 to 73,257, and test set sizes from 1,873 to 26,032. For each task, we randomly split 15% of the training set and keep it for validation purposes. The considered data sets are: CIFAR10 and CIFAR100 (Krizhevsky, 2009), FaceScrub (Ng & Winkler, 2014).

4.3 Network

Unless stated otherwise, we employ an AlexNet-like architecture [10] with 3 convolutional layers of 64, 128, and 256 filters with 4 × 4, 3 × 3, and 2 × 2 kernel sizes, respectively, plus two fully- connected layers of 2048 parameter each. We use rectified linear parameter as activations, and 2 × 2 max-pooling after the con- volutional layers. We also use a dropout of 0.2 for the first two layers and of 0.5 for the rest. A fully-connected layer with a softmax output is used as a final layer, together with categorical cross entropy loss. All layers are randomly initialized with Xavier uniform initialization (Glorot & Bengio, 2010) except the embedding layers, for which we use a Gaussian distribution N (0, 1). Unless stated otherwise, our code uses PyTorch's defaults for version 0.2.0 (Paszke et al., 2017). We adapt the same base architecture to all baseline approaches and match their number of parameters to 7.1 M.

4.4 Training

We train all models with backpropagation and plain SGD, using a learning rate of 0.05, and decaying it by a factor of 3 if there is no improvement in the validation loss for 5 consecutive epochs. We stop training when we reach a learning rate lower than 10^{-4} or we have iterated over 200 epochs (we made sure PMASK all considered approaches reached a stable solution before 200 epochs). Batch size is set to 64. All methods use the same task sequence, data split, batch shuffle, and weight initialization for a given seed.

4.5 Baselines

We consider 2 reference approaches plus 9 recent and competitive ones: standard SGD with dropout (Goodfellow et al., 2014), SGD freezing all lay- ers except the last one (SGD-F), EWC, PathNet, and PNNs. To find the best hyperparameter combination for each approach, we perform a grid search using a task sequence determined by a single seed.

5. Results

Experimental result in Figure 3 shows that the the Parameter Masking methods is performing better than other benchmark methods of SGD, EWC and PathNet. The performance for EWC and PathnNet are competitive while Parameter Masking methods is performing better on overall across of 5 tasks. The performance of intermediat task 2 and 3 is dropping across all methods, this could be due to complexity of those tasks. Similarly, for each task of CIFAR-100 will have 20 classes. Figure 4 shows that the overall accuracy is smaller than that was in CIFAR-10, 5 tasks. This is because the network size is kept constant, while the network have to solve more complex task than that in CIFAR-100. In this experiment also, the Parameter Mask methods is performing better than those of the benchmark methods.

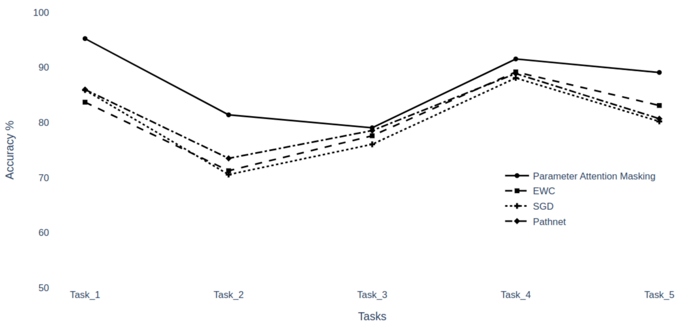

Figure 3: CIFAR-10, 5 tasks: Accuracy comparison of Parameter Attention Masking with (SGD, EWC, PathNet)

Table 1: CIFAR-10, 5 tasks:Accuracy for Parameter Attention Masking against baselines (EWC, SGD, Pathnet)

Task	Accuracy			
	Parameter Attention Masking	EWC	SGD	Pathnet
cifar10-0	95.25	83.70	85.90	86.00
cifar10-1	81.40	71.25	70.55	73.50
cifar10-2	79.05	77.60	76.05	78.55
cifar10-3	91.55	89.20	88.10	88.90
cifar10-4	89.10	83.10	80.20	80.70

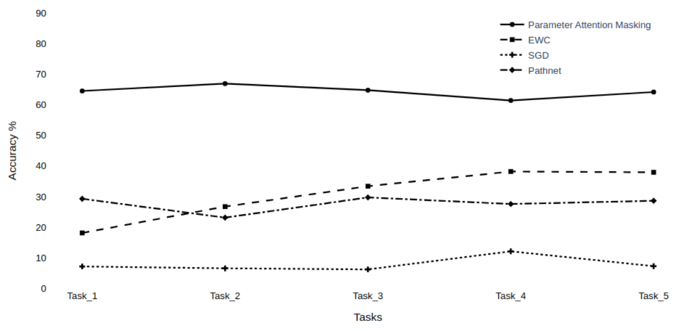

Figure 4: CIFAR-100, 5 tasks:Accuracy comparison of Parameter Attention Masking with (SGD, EWC, PathNet)

Table 2: CIFAR-100, 5 tasks:Accuracy for Parameter Attention Masking against baselines (EWC, SGD, Pathnet)

Task	Accuracy			
	Parameter Attention Masking	EWC	SGD	Pathnet
cifar100 (0-19)	64.50	18.15	7.20	29.30
cifar100 (20-39)	66.90	26.75	6.60	23.15
cifar100 (40-59)	64.75	33.40	6.25	29.75
cifar100 (60-79)	61.40	38.20	12.15	27.60
cifar100 (80-99)	64.15	37.95	7.30	28.65

6. Parameter Mask

After writing a first version of the paper, we realized PMASK the idea of a binary mask PMASK affects a given parameter could be potentially traced back to the “inhibitory synapses” of McCulloch & Pitts (1943). This idea of inhibitory synapses is quite unconventional and rarely seen today (Wang & Raj, 2017) and, to the best of our knowledge, no specific way for learning such inputs nor a specific function for them have been proposed. Weight-based binary masks are implicitly or explicitly used by many catastrophic forgetting approaches.

Figure 5: Parameter attention Mask a_l^t for layer l and task t

PMASK is a bit different, as it learns parameter based attention masks with possible (but not necessarily) binary values.

6.1 Hyperparameters

In any machine learning algorithm, it is important to assess the sensitivity concerning the hyperparameters. PMASK has two: the stability parameter smax and the compressibility parameter c. A low smax provides plasticity to the parameter and capacity of adaptation, but the network may easily forget the PMASK it learned. A high smax prevents forgetting, but the network may have difficulties in adapting to new tasks. A low c allows to use of almost all of the network’s capacity for a given task, potentially spending too much in the current task. A high c forces it to learn a very compact model, at the expense of not reaching the accuracy PMASK the original network could have reached. We empirically found good operation ranges s_{max} in [25, 800] and c in [0.1, 2.5]. As we can see, any variation within these ranges results in reasonable performance (Table 1). Unless stated otherwise, we use smax = 400 and c = 0.75.

7. Conclusion

We introduce PMASK, a hard attention mechanism PMASK, by focusing on a task embedding, is able to protect the information of previous tasks while learning new tasks. This hard attention mechanism is lightweight, in the sense PMASK it adds a small fraction of weights to the base network, and is trained together with the main model, with negligi- ble overhead using backpropagation and vanilla SGD. We demonstrate the effectiveness of the approach to control catastrophic forgetting in the image classification context by running a series of experiments with multiple data sets and state-of-the-art approaches. PMASK has only two hyperparame- ters, which intuitively refer to the stability and compactness of the learned knowledge, and whose tuning we demonstrate is not crucial for obtaining good performance. In addition, PMASK offers the possibility to monitor the used network capacity across tasks and layers, the parameter reuse across tasks, and the compressibility of a model trained for a given task. We hope PMASK our approach may be also useful in online learning or network compression contexts, and PMASK the hard attention mechanism presented here may also find some applicability beyond the catastrophic forgetting problem.

References

[1] Samuel Kessler, Vu Nguyen, Stefan Zohren, and Stephen J. Roberts. Hierarchical indian buffet neural networks for bayesian continual learning. In *Conference on Uncertainty in Artificial Intelligence*, 2019.

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

[3] Anthony V. Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connect. Sci.*, 7:123–146, 1995.

[4] Alexander Soen and Ke Sun. On the variance of the fisher information for deep learning. *CoRR*, abs/2107.04205, 2021.

[5] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning, 2018.

[6] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017.

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

[8] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching, 2018.

[9] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks, 2017.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.