International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Efficient and Effective Architecture in Continual Learning Through Various ResNets

Sachin Gaur[1], Rahul Pandey[2]

Submitted: 25/01/2024 **Revised**: 03/03/2024 **Accepted**: 11/03/2024

Abstract: Continual learning stands as a crucial component in advancing artificial intelligence, yet it encounters a significant challenge known as catastrophic forgetting. This phenomenon occurs when models lose previously acquired knowledge upon learning new tasks. While some methods propose partial remedies, the impact of altering the model's architecture on this forgetting remains largely unexplored. This study delves into Residual Networks (ResNets) to evaluate how modifications in depth, width, and connectivity influence the process of continual learning. By introducing a simplified design tailored specifically for continual learning, this research seeks to compare its efficiency against established ResNets. Through an in-depth exploration of the algorithm's configuration, the study aims to elucidate the underlying rationale behind its design decisions. Furthermore, it evaluates the performance of the proposed model using a diverse set of metrics, aiming to identify both strengths and areas for improvement. Ultimately, this research sheds light on how the architectural aspects of a model impact its learning capabilities over time, with the ultimate goal of fostering the development of AI systems capable of continuous learning without experiencing the detrimental effects of forgetting. Demonstrating accuracy levels ranging from 62.52% to 90.39% across various tasks, the proposed model showcases its effectiveness in real-world continual learning scenarios.

Keywords: *Catastrophic forgetting, Continual learning, ESPN, Machine learning, ResNet*

1. Introduction

In machine learning algorithms and human intelligence, human intelligence can perform better in many tasks, whereas most of the machine learning models are created for one task, and by using the same models for multiple tasks, their accuracy decreases significantly Silver et al.[1-2]. Continual learning or learning without forgetting aims to build models that are capable of multitasking which are never-ending sequences of tasks [3]. While in these models that are capable of continual learning the main problem that arises is catastrophic forgetting [4]in which when a model tries to learn a new thing it significantly forgets the old learnings from old tasks. Numerous research studies have delved into the field of continuous learning, investigating techniques such as parameter isolation and expansion-based approaches. These methods primarily focus on algorithms with a fixed architecture, aiming to enhance the adaptability and long-term performance of artificial intelligence systems. Parameter isolation techniques safeguard previously acquired knowledge by isolating specific model parameters while accommodating new information. Conversely, expansion-based methods [5] seek to increase the model's capacity to assimilate new knowledge without disrupting existing knowledge. Despite the diverse range of approaches in continuous learning research, there remains a need to explore dynamic architectures that seamlessly evolve and adapt to changing tasks and environments. Such advancements hold promise for addressing the challenges posed by catastrophic forgetting, allowing AI systems to continually learn and improve their performance over time. There are various real-life use cases and decision-making scenarios that support the advancement of continual learning [6]. Most algorithmic approaches use the Resnet18 architecture model [7] because residual networks (ResNets) redefine the layers by learning residual functions for the input at each layer. Instead of directly learning the desired transformation, ResNets focus on understanding the differences (residuals) between the input and output. These residual blocks incorporate skip connections, allowing the network to learn identity mappings when necessary. The Split CIFAR-100 and ImageNet [8-9] datasets are commonly employed in research, and Split CIFAR-100 is divided into 20 distinct, non-overlapping subsets. Each of these subsets serves as a separate training or evaluation set, allowing for more comprehensive analysis and experimentation. Biological intelligence is still far superior to ANN in multi-task learning. ANN models are created for specific tasks only, but by increasing the domain of learning tasks in ANN, we can efficiently use the resources and flexibility comparable to human learning so that it can perform in dynamic and unpredictable environments [10-11]. Several alternative approaches, such as replay-based methods and regularization techniques, leverage valuable information from previously trained models. These methods aim to mitigate issues like catastrophic forgetting when learning new tasks. The proposed model, in contrast, builds upon the ESPNET [12] algorithm, which represents an

[1] *B.T Kumaon Institute Of Technology Dwarahat, INDIA*
ORCID ID : 0000-0002-7638-3875
[2] *B.T Kumaon Institute Of Technology Dwarahat, INDIA*
ORCID ID : 0009-0002-5530-9729
** Corresponding Author Email: ersgaur1234@gmail.com*

evolution beyond the earlier Packnet [13]. By incorporating advancements from ESPNET, the proposed model enhances its ability to handle continual learning scenarios effectively. It has been found that continuous learning is similar to learning in animals [14]. In response to the challenge of unsupervised embedding learning, efficient similarity measures between samples within a low-dimensional embedding space [15] was proposed. This research's main aim is to work on creating an architecture model that can perform continual learning efficiently and effectively by using the algorithm that is best suited for our architecture. Our goal is to create an architecture that reduces catastrophic forgetting without a decrease in the accuracy of overall model performance. the research has tested the proposed model against various other models and found that the proposed model produces the best results among them. To create such a model, we first need to understand the important parameters that may affect the proposed model, such as the width and depth of the model, and how they affect the learning and retention rate.

2. Related Work

Various methods had been proposed to solve the problem of catastrophic forgetting which can be categorized into three groups regularization-based continual learning, which is developed by a second-order Taylor approximation of the loss function of individual tasks by Lopez-Paz et al.[16] and Yin et al.[17]replay-based, which greatly affects the performance of continual learning using experience replay (ER) Buzzega et al. [18] and architecture, which shows how the quadratic regularization technique plays an important role in architecture to create a model suitable for continual learning. James et al. [19]. Our main focus is to minimize the problem of catastrophic forgetting with the help of the architecture method using different sets of ResNets networks. algorithmic work such as adding three fine-grained classification tasks into a single ImageNet-trained VGG-16 network and accomplishing accuracies near those of individually trained networks for each task using the Packnet algorithm Fernando et al. [20].

Super mask in superposition (SupSup) can provide better results using gradient-based optimization to encounter an unbent superposition of learned super masks, which supports minimizing the resulting entropy Wortsman et al. [21]. The method of continual learning uses memory-based solutions such as rehearsal or pseudo-rehearsal. This strategy involves storing prototype instances of past tasks in episodic memory and replaying them during training to prevent forgetting. Knowledge Distillation and Feature Learning (ICARL) Rebuffi et al. [22] ICARL combines knowledge distillation and feature learning by storing examples closest to the feature mean of each class in fixed memory and using distillation loss to mitigate forgetting. Gradient Episodic Memory (GEM) and Average-GEM (AGEM) Hou et al. [23] These methods employ an inequality constraint via episodic memory to prevent forgetting past tasks. Unified Classifier with a Cosine Linear Layer Chaudhry et al. [24] and Wu et al. [25] proposes learning a unified classifier using a cosine linear layer and then progressively acquiring new knowledge. Sample-based techniques utilize Pham et al. [26] samples to create an additional validation set, enhancing model generalization. Generative Models (Auto-encoder, VAE, GAN) (Bengio et al., Kingma et al., Goodfellow et al.)[27, 28, 29] Generative models are employed to replicate samples from past tasks. By replaying generated data, these models prevent forgetting (Wu et al., Shin et al., Ostapenko et al.) [30, 31, 32]. These methods collectively address both catastrophic forgetting and retention, ensuring a more robust learning process.

2.1. Algorithmic work

On the aspect of algorithms, various methods have been proposed to reduce the problem of catastrophic forgetting. Such as Apiece Synapse, which gathers task-appropriate input over time and uses this report to rapidly store new memories without ignoring old ones. This method approaches the continual learning of classification tasks and establishes that it reduces forgetting while preserving computational efficiency Zenke et al.[33] Meta- techniques of meta-learning and replay-based learning to optimize the model in the aspects of transfer and interference Chaudhry et al. [34] experience replay (MER) is a method that combines the Even the tiniest memory or episodic memory from the past increases the generalization of future tasks, which increases the overall performance of the machine learning model. Other algorithms help in adaptive learning, in which learning rates are automatically adjusted and show more yield in RBMs (Restricted Boltzmann Machines) Cho et al. [35]. A task-based hard attention method has been proposed that saves the information from the previous task without affecting the learning of the current task Serra et al. [36].

2.2. Architecture work

Various architectural methods are also proposed to solve the problem of catastrophic forgetting, such as the expectation-maximization (EM) method, which automatically selects the appropriate transfer configuration and also optimizes the network weights related to each task Lee et al. [37]. Network quantization and pruning to learn binary masks that piggyback on existing networks to deliver good arrangements on a new task Mallya et al. [38]. Other methods like this consist of two parts: neural structure optimization and fine-tuning. By separating the structure learning and the parameter estimation, this method can perform better in many ways Li et al.[39]. By performing iterative pruning and network retraining, we can pack multiple tasks into a single network with the help of a packet

algorithm. Deep neural network architecture, which can expand and adjust dynamically for lifelong learning, is a great advancement in continual learning, but it is still less accurate than other pre-trained models Yoon et al. [40].

3. Proposed Architectures

This paper will present a rigorous investigation of the algorithm's configuration and structure, performing an in-depth exploration of the subtle details and complexities that form the foundation of the proposed method. The paper will focus on a thorough analysis of the various elements, parameters, and decisions that have been deliberately designed to enhance the efficiency and effectiveness of the architecture. Moreover, it is crucial to not only explain the technical aspects of the proposed structure but also reveal the underlying principles and ideologies that have guided its development. This comprehensive perspective will offer insight into the extensive reasoning and considerations that have influenced the creation of the paper's unique solution. Furthermore, examining the diverse set of metrics that have been selected for evaluating and comparing structure. These metrics have been carefully chosen to provide a multifaceted evaluation of our algorithm's performance, covering factors such as accuracy, precision, scalability, and robustness. This paper's goal is to emphasize the thorough and rigorous nature of our evaluation process through a comprehensive analysis of these metrics. Finally, will perform a deep-dive analysis of the results produced by our algorithm. This analysis will go beyond mere numerical values, probing into the identification of insights, trends, and patterns within the data. This paper aims to interpret these results in a way that not only showcases the strengths of our structure but also identifies areas for potential improvement and further optimization. Fig.1. illustrates the operational framework outlined in this research. the research uses the CIFAR100 dataset, which consists of 6,000 training images across 100 different classes. The proposed method described in this study entails selecting 20 random classes to compose a single set, resulting in a total of 20 distinct sets derived from the CIFAR100 dataset. By incorporating weights from previous tasks, the proposed architecture integrates them with the training of new tasks, thereby harnessing the advantages of continual learning.

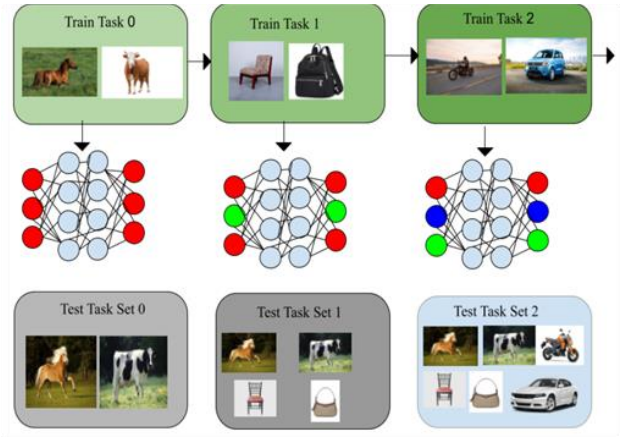

Fig. 1. Working of Proposed Architecture

3.1. Setup

We evaluate our model on various metrics using the CIFAR100 dataset. This study goal is to measure how our model performs in terms of accuracy and hardware cost. the research wants to show that the proposed model is both efficient and effective.

3.2. Architecture Setup

The proposed architecture is the modification of GemResnet18 in terms of width and depth and created various versions of ResNet along with the algorithm of ESPN. We name this networks as VResNet1, VResNet2, VResNet3, VResNet4.

Width per group refers to the number of channels or feature maps produced by each group of convolutional layers in a neural network. In the context of ResNet and similar architectures, the network is divided into groups, and each group contains convolutional layers responsible for extracting certain features from the input data. The width per group specifies how many feature maps are generated by each group of layers. For example, if a network has a width per group of 20, it means that each group produces 20 feature maps or channels as output. The term "basic block configuration" refers to how layers are organized within a fundamental unit of a convolutional neural network, such as ResNet. In this context, it specifies the number of convolutional layers arranged within each block. For instance, a configuration like (1, 4, 4, 1) implies that the block comprises four convolutional layers, with fewer filters in the first and last layers compared to those in between. Similarly, configurations such as (2, 8, 8, 2) and (4, 1, 1, 4) indicate variations in layer count and filter sizes within the basic block. These arrangements influence the network's ability to capture features at different levels of abstraction, determining its complexity and capacity. VResNet1 has 20 widths per group and has basic blocks of (1,4,4,1). VResNet2 has 80 widths per group and has basic blocks of (2,8,8,2). VResNet3 has 80 widths per group and has basic blocks of (2,1,1,2). VResNet4 has 80 widths per group and has basic blocks of (4,1,1,4). GemResNet18 [29] has 20

widths per group and has a basic block of (2, 2, 2).

In Fig.2. the flowchart outlines the process of modifying the GemResNet18 architecture and creating variants like VResNet1-4 with adjusted width and depth. It then details the steps for training using the ESPN algorithm, including weight initialization, pre-training, pruning, and fine-tuning. Hyperparameters such as dataset, batch size, optimizer, and learning rate are specified, along with pruning iterations and FLOPs values. Evaluation metrics like average accuracy, learning accuracy, average loss, average forgetting, and FLOP count are also defined. The

Flowchart starts with modifying the architecture, moves on to setting up the algorithm, defining hyperparameters, and finally

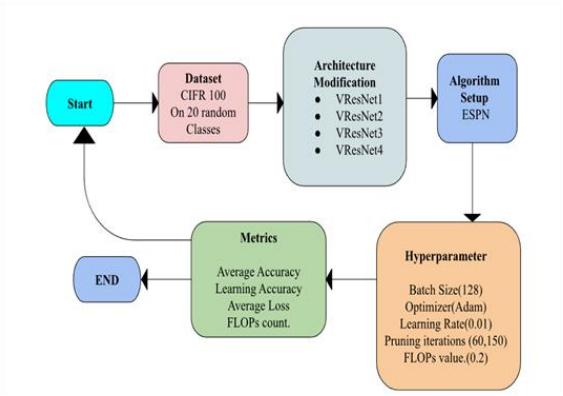

Fig.2. Flowchart of architecture

outlining evaluation metrics. Each step is clearly defined and connected, guiding the process from architecture modification to evaluating model performance.

3.3. Algorithmic Setups

The proposed models are trained using the ESPN algorithm. This algorithm is particularly lauded for its ability to curtail the number of floating-point operations (FLOPs), thereby resulting in substantial time savings throughout the training process. At the outset, the subnetwork is initialized with zero tasks, indicating that all weights are set to zero as a starting point. It adopts a task sequence T along with a model weight w, with tasks incrementally added in sequence. The primary objective is to pinpoint the most suitable sub-network that achieves the dual goals of FLOP reduction and sparsity enhancement without compromising the overall performance of the model. To achieve this objective, the study leverages a combination of joint channel and weight pruning methodologies. This strategic approach not only enhances the efficiency of the models but also ensures their effectiveness across a diverse array of tasks and scenarios. By integrating these pruning techniques into the training process, the study aims to advance the capabilities of continuous learning frameworks and pave the way for further future innovations. The provided algorithm offers a structured approach for continual learning tasks. It begins by establishing task masks, denoted as M, to encapsulate the parameters associated with each task in the learning sequence. Additionally, a set of new weights, m_new, is initialized, typically with a predefined value. Subsequently, the algorithm iterates through the task sequence, performing the following steps for each task: During the pre-training phase, the model parameters are refined through iterative updates to the task mask M. These updates involve subtracting a fraction of the gradient of the loss function for the model parameters, computed using the new weight set m_new. At the onset of each task, a sub-network mask, m_t, is initialized, often mirroring the weights of m_new. Following this, the algorithm enters a loop for gradual channel and weight pruning, persisting until both FLOPs and weight sparsity constraints are met. Within this loop, the task mask M undergoes updates via gradual channel and weight pruning techniques. This entails subtracting a portion of the gradient of the loss function from the model parameters, computed using the intersection of the sub-network mask m_t and the new weight set m_new. Once the pruning constraints are satisfied, the algorithm proceeds to fine-tune. Here, further updates to the task mask M are made by subtracting a fraction of the gradient of the loss function for the model parameters, computed using the intersection of the sub-network mask m_t and the new weight set m_new. Upon completing fine-tuning for each task, the task mask set M is refreshed to include the current sub-network mask m_t additionally, the new weight set m_new is adjusted by removing parameters corresponding to the current sub-network mask m_t.

Algorithm. Efficient Sparse PackNet (ESPN-1)

Require variables. Task Sequence = T, model weight= M, step_size = N, pre-training, pruning, fine-tuning duration; weight allocation parameter = @.

1. Set of task mask M
2. Set of new weight mnew=[p]
3. for t in T do:
   while pre-training do:
M = M – N *(∇LSt (θ) mnew)
End while
4. Initialize sub-network mask m_t=[p]
5. while FlOPs(1) & weight_sparsity(@) constraints not satisfied do:
   Gradual channel & weight pruning: update m
   M = M – N *(∇LSt (θ) (m_t ∩ m_new))
   end while
6. while fine_tuning do:
   M = M – N*(∇LSt (θ) (mt ∩ m_new))
  end while
7. M=M ∪ {t: m_t}
8. m_new = m_new \ m_t
9. end for

3.4. Hyperparameters

The proposed architecture trains different models on hyperparameters and uses a dataset of Ciffar100, where the proposed architecture uses the batch size 128 and Adam optimizer with (β1, β2)=(0, 0.999) and a learning rate of 0.01 with 20 number epochs. The pruning iteration is a matrix of [60, 150] with a Flops value of 0.2. With the vanilla L2 loss regularization, we minimize the error in our continual learning task [41].

3.5. Metrics

This research chose different types of metrics to prove the accuracy of the experiments. This study defines the metrics on two bases: (1) how well the proposed architecture performs on learning, and (2) how well the proposed architecture performs based on efficiency. For the former, this research records the average accuracy, learning accuracy, joint/multi-task accuracy, and also the average forgetting of the model. We also take the average FLOPs count.

Average Accuracy (0–100) is a metric that shows how well a model or system performs on a set of X tasks based on their validation accuracies. The higher the average accuracy, the better the model or system. We can define some variables as follows: X: The total number of tasks. a(i). The validation accuracy of the i'th task. The formula can be written as

$$Q = \frac{1}{X} \cdot \sum_{i=0}^{X} a(i) \quad (1)$$

Q is the "average validation accuracy." It is the average of the validation accuracies of all the tasks in the set. 1/X: This term is used to get the average value. It divides the total sum of the validation accuracies by the number of tasks, X. Σ(i=0 to X): This is a sum (sigma) symbol that tells us to add up the following expression for all values of i from 0 to X. In other words, we are adding up the validation accuracies of all tasks, from the first one (i = 0) to the last one (i=X). a(i): This is the validation accuracy of the i'th task. It shows how well the model or system does on each task. To sum up, the formula calculates the average validation accuracy of a model or system across all X tasks by adding up the validation accuracies of each task and then dividing by the number of tasks, X. The result, Q, can be used as a measure of the model's average performance on the set of tasks, with higher values indicating better average accuracy. This can be a useful metric for evaluating the overall effectiveness of a model across multiple tasks.

Learning accuracy (0–100) is a metric that shows how well a learner or system does on X tasks. It also reflects the plasticity or adaptability of the learner or system. The higher the learning accuracy, the better the learner or system. It is calculated as

$$LQ = \frac{1}{X} * \sum_{i=0}^{X} a(i,i) \quad (2)$$

LQ, or "learner's quality," is a score or measure. It evaluates the overall quality or performance of a learner or system involved in X tasks; 1 / X is the term used to get the average value. It divides the total sum of the accuracies by the number of tasks, X. Σ(i=0 to X) is a sum (sigma) symbol that tells us to add up the following expression for all values of i from 0 to X. In other words, we are adding up the accuracies of all tasks, from the first one (i = 0) to the last one (i= X), where a(i, i) is the accuracy of the i'th task. It shows how well each task is done. The formula calculates the average accuracy or quality of a learner or system across all X tasks by adding up the accuracies of each task and then dividing by the number of tasks, X. The result, LQ, can be used as a measure of how well the learner or system performs on a set of tasks, with higher values indicating better performance.

$$L2regularization = \lambda * (/w/)^2 \quad (3)$$

Average Loss (0 to 100): This metric shows how much error or deviation a model or system has on a set of tasks. The lower the loss function, the better the model or system. We are using l2 regularization to reduce the error of the model. Where λ (lambda) is the regularization parameter, controlling the strength of regularization and w is the vector of model weights.||w||^2 represents the squared L2 norm of the weight vector.

Average Forgetting (-100 to 100) is metric shows how much a model or system forgets its previous learning while learning new tasks. (Loss on Task i: Best Previous Loss) N represents the total number of tasks. Loss on Task i (L_i) is how much error or deviation the model or system has on the ith task. The Σ from i = 1 to N indicates that we need to add up all tasks. The formula is defined as

$$AF = (1/N) * \sum_{i=1}^{N} (L_i - BestPreviousLoss) \quad (4)$$

Best Previous Loss is the lowest error or deviation observed on any of the previous tasks from task 1 to task i-1. This formula calculates the average forgetting by finding how much worse (higher loss) the model or system does on each task compared to its best performance (lowest loss) on any earlier task. The result shows how much forgetting occurs on average when learning new tasks.

3.6. Result and Analysis

The proposed architecture results from the average accuracy of different models with 20 random tasks from the CIFAR100 dataset. CIFAR-100 is a dataset of 60,000 colored images of size 32x32, each belonging to one of 100 fine-grained classes, such as 'apple',

'Bee', 'rose', 'castle', and so on. These classes are also organized into 20 coarse-grained superclasses, creating a hierarchical structure for the dataset. For instance, the

superclass ‘fish’ includes the classes ‘aquarium fish’, ‘whale', 'shark’, and ‘ray’. CIFAR-100 is a difficult dataset for image classification tasks, especially for models that aim to identify a wide range of objects in natural scenes. This study researched how well the architecture performs on different tasks.

Table 1. Accuracy measures of all architecture compared to different tasks

Task Number	VResnet 1 Accuracy	VResnet 2 Accuracy	VResnet 3 Accuracy	VResnet4 Accuracy
1	45.4	57.6	59.4	57.4
2	68.2	73.8	72.8	71.2
3	66.0	72.0	73.4	73.6
4	74.1	76.4	79.4	70
5	70.6	72.4	81.6	83
6	67.6	82.6	83.2	85.6
7	66.2	67.4	72.8	78.0
8	72.0	70.4	80.6	60.2
9	68.4	79.4	72.4	71.2
10	59	82.0	85.0	82.2
11	66.8	69.4	76.0	79.6
12	71.8	69.0	78.2	65.8
13	60.6	71.0	41.2	77.8
14	59.2	72.0	68.4	70.0
15	39.2	50.2	44.8	39.6
16	60.8	54.6	58.8	63.2
17	57.4	60.0	57.6	49.0
18	59.4	66.0	61.8	63.4
19	48.6	42.2	47.2	79.0
20	69.0	90.4	77.0	80.8

Table 1. Provides information about the number of tasks and the corresponding accuracy scores for different versions of the ResNet model, namely VResNet1, VResNet2, VResNet3, and VResNet4. The "Number of Tasks" column represents the random tasks that are taken from the dataset of CIFAR100. In the subsequent columns, we have the accuracy scores for each version of the ResNet model. For instance, vResNet1 achieved a 72% accuracy score for task 7, VResNet2 81.99% score for task 9, and so on. These accuracy scores indicate how well each version of the ResNet model performed on the given tasks. The number of tasks

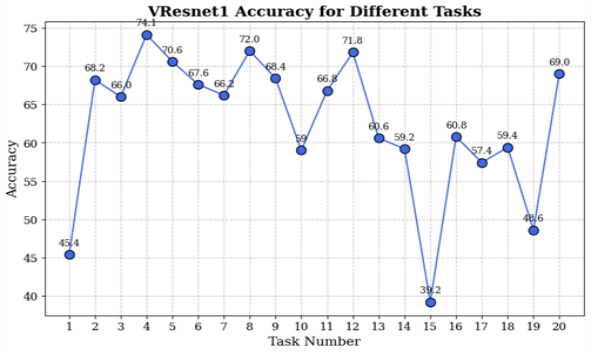

Fig.3. Accuracy of VResNet1

could vary, and these accuracy scores provide valuable insights into the models' performance, helping us understand which version of ResNet may be more suitable for the tasks at hand.

In Fig.3. The graph depicts the relationship between the number of tasks (x-axis) and the accuracy of the vResNet1 model (y-axis).

The x-axis, labelled as "Task Number to be evaluated," represents the sequential order of tasks, starting from 0 and going up to 19. Each point on the x-axis corresponds to a specific task. The y-axis, labelled as "Accuracy," represents the accuracy scores achieved by the VResNet1 model for each task. The accuracy values are represented as percentages. At the beginning, for task 0, the accuracy is around 45.4%.The accuracy generally increases as we move along the x-axis, reaching a peak of approximately 74.2% around task 3.

There are some fluctuations in accuracy scores as we progress through the tasks, but an overall increasing trend is visible. Towards the end, the accuracy stabilizes at a level around 68-69%. This graph provides a clear visual representation of how the accuracy of the VResNet1 model changes as it is evaluated on an increasing number of tasks. It is evident that, in general, as more tasks are evaluated, the model's accuracy tends to improve, although there may be fluctuations along the way.

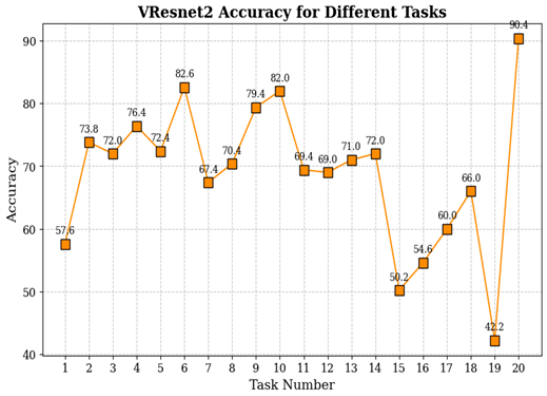

Fig.4. Accuracy of VResNet2

In Fig.4. The graph illustrates the relationship between the

number of tasks (x-axis) and the accuracy of the VResNet2 model (y-axis). The x-axis, labeled as "Task Number to be evaluated," represents the sequence of tasks from 0 to 19, indicating each task's order in evaluation. The y-axis, labeled as "Accuracy," presents the accuracy scores attained by the VResNet2 model for

Each corresponding task, represented as percentages. Initially, for task 0, the accuracy stands at around 57.6%. The accuracy consistently improves as we progress through the tasks, reaching a peak of roughly 90.4% around task 19. The VResNet2 model displays substantial accuracy gains during the evaluation of tasks. While there might be minor fluctuations along the way, the general trend shows a notable increase in accuracy. This graph provides a visual representation of how the accuracy of the VResNet2 model evolves as it is evaluated across an increasing number of tasks.

In Fig.5. The graph depicts the relationship between the number of tasks (x-axis) and the accuracy of the VResNet3 model (y-axis). The x-axis, labeled as "Task Number to be Evaluated," represents the sequence of tasks from 0 to 19, indicating the order in which each task is evaluated. The y-axis, labeled as "Accuracy," shows the accuracy scores achieved by the VResNet3 model for each respective task, represented as percentages. At the outset, for task 0, the accuracy is approximately 59.4%.

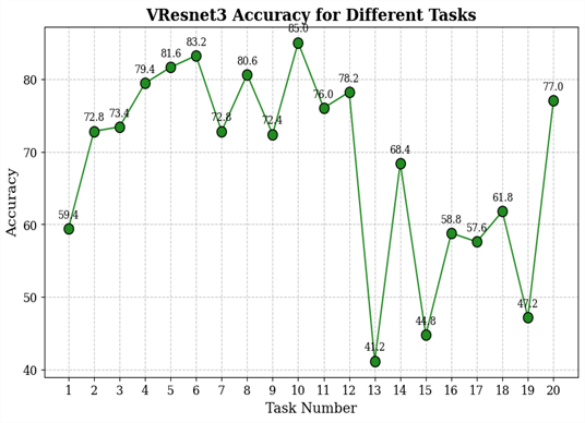

Fig.5. Accuracy of VResNet3

The accuracy shows a consistent upward trend as the number of evaluated tasks increases, with notable gains. The model's performance steadily improves, reaching a peak of about 85.0% around task 9. While there may be minor fluctuations, the overall trend suggests that the VResNet3 model becomes more accurate as it is evaluated on additional tasks.

This graph visually represents how the accuracy of the vResNet3 model evolves as it is assessed across an increasing number of tasks.

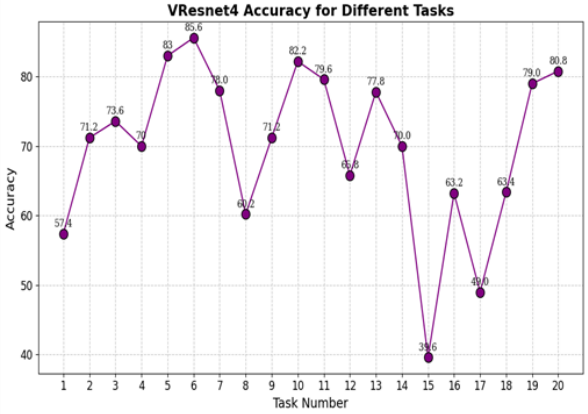

Fig.6. Accuracy of VResNet4

In Fig.6. the graph illustrates the relationship between the number of tasks (x-axis) and the accuracy of the vResNet4 model (y-axis). The x-axis, labelled as "Task Number to be Evaluated," represents the order of tasks from 0 to 19, indicating the sequence of evaluation. The y-axis, labelled as "Accuracy," represents the accuracy scores achieved by the vResNet4 model for each respective task, presented as percentages. Initially, for task 0, the accuracy is approximately 57.4%.The accuracy exhibits fluctuations as the model is evaluated across tasks, with both increases and decreases. Notable peaks in accuracy are observed at different points, with the highest accuracy of around 85.6% occurring around task 5. The model's performance shows variability as it is evaluated on different tasks.

This graph visually illustrates the accuracy of the VResNet4 model changes as it is assessed across an increasing number of tasks. It suggests that the model's accuracy can vary considerably across tasks, with some tasks generating significantly higher accuracy than others.

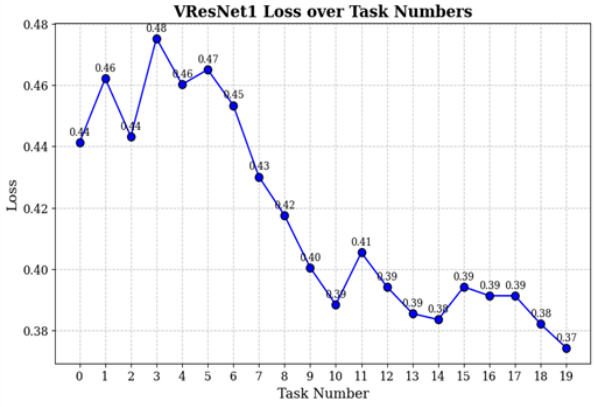

Fig.7.VResNet1 Loss over Task Numbers

In Fig.7. The graph represents the relationship between the number of tasks (x-axis) and the L2 loss function output for the VResNet1 model (y-axis). The x-axis, labelled as "Task Number to be Evaluated," indicates the sequence of tasks being evaluated, ranging from 0 to 19.The y-axis, labeled as "L2 Loss Function Output," reflects the values of the L2 loss function, which is a measure of the error or discrepancy

between the model's predictions and the actual target values. Lower values on the y-axis indicate a better fit of the model to the data. At the beginning, for task 0, the L2 loss is approximately 0.4413.The L2 loss generally increases as more tasks are evaluated, indicating a rise in the discrepancy between predictions and actual values. The loss function experiences fluctuations as it is evaluated on different tasks. Towards the end, there is a tendency for the L2 loss to stabilize, with values around 0.3743 to 0.4752.

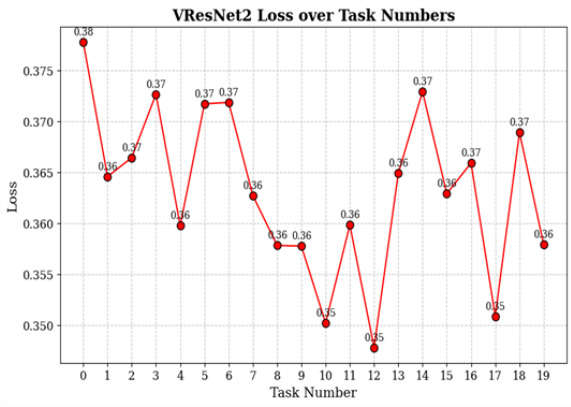

Fig.8.VResNet2 Loss over Task Numbers

It indicates how well the model fits the data and provides insights into the model's performance in minimizing prediction errors for different tasks.

In Fig.8. The graph illustrates the relationship between the number of tasks (x-axis) and the accuracy of the VResNet4 model (y-axis). The x-axis, labeled as "Task Number to be Evaluated," represents the order of tasks from 0 to 19, indicating the sequence of evaluation. The y-axis, labeled as "Accuracy," represents the accuracy scores achieved by the VResNet4 model for each respective task, presented as percentages.

Initially, for task 0, the accuracy is approximately 57.4%. The accuracy exhibits fluctuations as the model is evaluated across tasks, with both increases and decreases. Notable peaks in accuracy are observed at different points, with the highest accuracy of around 85.6% occurring around task 5. The model's performance shows variability as it is evaluated on different tasks. This graph visually represents how the accuracy of the vResNet3 model evolves as it is assessed across an increasing number of tasks. It demonstrates that the model becomes increasingly effective in handling diverse tasks, with accuracy improving as more tasks are evaluated.

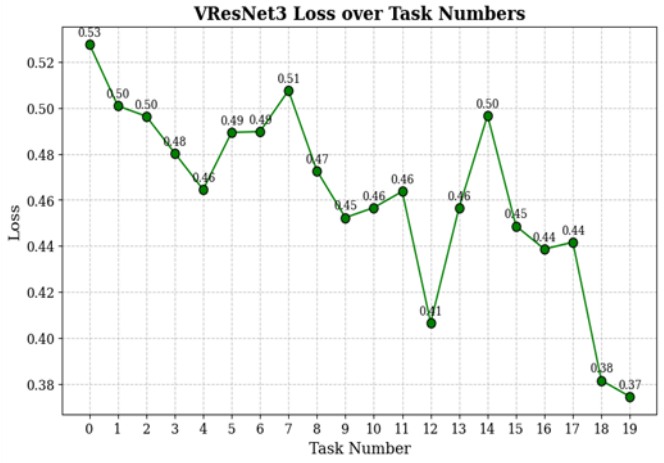

Fig.9.VResNet3 Loss over Task Numbers

In Fig.9. The graph depicts the relationship between the number of tasks (x-axis) and the loss values for the VResNet3 model (y-axis). The x-axis, labelled as "Task Number to be Evaluated," represents the order in which tasks are evaluated, ranging from 0 to 19.The y-axis, labelled as "Loss," indicates the values of the loss function. In this context, the loss values represent the error or discrepancy between the model's predictions and the actual target values. Higher values on the y-axis suggest a greater divergence between the model's predictions and the actual data, signifying a less accurate model. At the start, for task 0, the loss value is around 0.5277. As the model is evaluated on different tasks, the loss values generally decrease, which implies improved model performance and a better fit to the data. The loss values continue to decrease as more tasks are evaluated, indicating the model's enhanced accuracy. This graph visually demonstrates how the loss, which reflects model performance, changes as the VResNet3 model is assessed across an increasing number of tasks. It shows the model's ability to minimize prediction errors and its overall performance in handling various tasks, with lower loss values indicating better accuracy.

In Fig.10. The graph illustrates the relationship between the number of tasks (x-axis) and the accuracy of the VResNet4 model (y-axis). The x-axis, labeled "Task Number to be Evaluated," indicates the order in which tasks are evaluated, ranging from 0 to 19. The y-axis, labeled "accuracy," represents the accuracy scores achieved by the VResNet4 model for each respective task, Presented as percentages. Initially, for task 0, the accuracy is approximately 57.4%. The accuracy experiences fluctuations as the model is evaluated on different tasks, with both increases and decreases. There are notable peaks in accuracy at different points, with the highest accuracy of around 85.6% occurring around task 5. The model's performance shows variability as it is evaluated for different tasks. This graph visually depicts how the accuracy of the vResNet4 model changes as it is assessed across an increasing number of tasks. It suggests that the model's accuracy can vary considerably

across tasks, with some tasks yielding significantly higher accuracy than others.

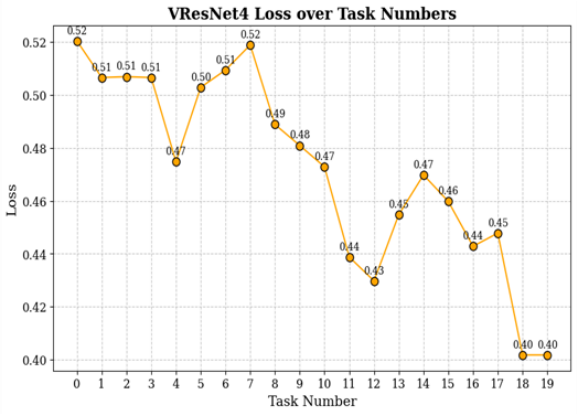

Fig.10.VResNet4 Loss over Task Numbers

Table 2. Architectures and their average loss tasks.

Architecture name	Average Loss
VResNet1	1.5045
VResNet2	1.56
VResNet3	1.286
VResNet4	1.5045

Table 2. From the above result we can easily see that VResNet4 contain the highest average accuracy among all other models and VResNet3 also has as good accuracy as compared to VResNet4 they both have the same number of weights but a different number of layers. From our result, we can say that shape and weight matter most for less catastrophic forgetting architected. VResNet2 have the largest number of layers among all the architecture but still produces less accuracy due to high catastrophic forgetting hence we can conclude that increasing the weight of architecture will increase accuracy and increasing the layers of architecture will have less effect on the accuracy for multiple tasks.

4. Architecture matter

Important elements of the proposed architecture's that are important for an efficient method of continuous learning are the pooling layer and global pooling layer, the width and depth of the architecture, and the effect of shape.

4.1. Effect of Shape

According to the result of this research experiment, this study found out that the U shape network performs well as compared to other models in terms of continual learning. We can see that Among GemResNet18 and VResNet3 the shape of VResNet3 is U with [2,1,1,2] stacks of layers this type of architecture has more number width per layer and more number of layers in the starting and end of the model. Which produces a significant increase of 4% in overall accuracy. Whereas in VResNet1 with [1, 4, 4, 1] stack of layers, the accuracy decreases significantly.

4.2. Role of Width and Depth

The work by Zhou et al. [42] suggests a simple change in the global average pooling layer and a class activation mapping technique that enable the network to both identify the image and locate the regions that belong to each class in one forward pass. The work demonstrates that the proposed method attains the best results on weakly supervised object localization on the ImageNet Large Scale Visual Recognition Challenge ILSVRC benchmark. It has been shown that with an increase in the width of architecture, the accuracy increases with the increase of related tasks, and according to the result of our experiment, as the width of architecture increases, the architecture performs much better as compared to an increase in depth. This study compares GemResNet18 and VResNet3 because same number of parameters. Here, VResNet3 has fewer layers or depth with a width per layer of 80, whereas GemResNet18 has a width per layer of 20 with more layers, and still, VResNet3 can produce a better average accuracy compared to GemRestNet18.

4.3. Polling layers

Lin et al. [43] proposed an elements in convolutional neural networks (CNNs) that help in decreasing the spatial dimensions of feature maps while keeping the relevant information. These layers are usually applied after convolutional layers to gradually reduce the spatial size of the representation, which aids in avoiding overfitting and also lowers computational complexity. There are different kinds of pooling layers, but the two most common ones are max pooling and average pooling. In max pooling, for each region of the input feature map, the output is the maximum value. This operation effectively keeps the most prominent features within each region, helping to maintain important information while decreasing spatial dimensions. In contrast to max pooling, average pooling computes the mean value within each region of the input feature map. This operation estimates the average activation of features, providing a smoother downsampling than max pooling. Pooling layers assist in achieving translation invariance, meaning the network is less affected by the exact location of features in the input. Moreover, they reduce the number of parameters and computational complexity in the network, which can prevent overfitting and make the model more computationally efficient. According to the result of the research experiment, the study compared the accuracy of architecture with 20 epochs and found that using max-pooling in architecture produces a good result as compared to the average pooling layer. From the observation of the results of our experiments, there is a 2-3% increase in the

average accuracy of the result by using the max pooling layer. It can be evaluated in Table 3.

Table 3. Accuracy on Average pooling layer vs Max pooling layer.

Architec-ture	Accu-racy on Average pooling layer	Accu-racy on Max pooling layer
VResNet1	62.52%	64.20%
VResNet2	68.90%	69.60%
VResNet3	68.58%	70.20%
VResNet4	69.78%	71.68%

4.4. Global Pooling Layers

Sermanet et al. [44] proposed a technique that reduces the size of feature maps in convolutional neural networks (CNNs) that are used for tasks such as image classification. It preserves the essential information in the feature maps while decreasing the spatial dimensions. In a CNN model, an input image goes through several convolutional and pooling layers, which produce feature maps that capture the spatial information of various features detected by the model. However, before making predictions, the feature maps need to be transformed into a fixed-length vector that can be inputted into a fully connected layer for classification or regression. This is the role of the global average pooling layer. It differs from traditional pooling layers like max pooling, which choose the maximum value in each region by computing the average of all values in each feature map. It effectively turns each feature map into a single value, thus reducing the spatial dimensions to 1x1. The global average pooling layer is simple to implement. For each feature map, the layer calculates the mean of all values in that map. This results in a vector where each element corresponds to the average activation of a specific feature over the entire spatial extent of the feature map. One of the main benefits of global average pooling is that it provides translation invariance, meaning the network is less affected by the exact location of features in the input image. Moreover, global average pooling reduces the number of parameters in the network, which can help avoid overfitting, especially in situations with limited training data. The study tested it with different proposed models and found that the global pooling layer causes more forgetting in between layers which henceforth decreases the accuracy of the model, which can have evaluated from Table 4.

Table 4. Effect of Global pooling layer vs without Global pooling layer.

Architectu re	Accuracy with GPL	Accuracy without GPL
VResNet1	59.78%	62.52%
VResNet2	64.57%	68.9%
VResNet3	65.37%	68.58%
VResNet4	67.67%	69.78%

5. Comparative analysis

The comparative analysis of all the proposed architecture concerning their accuracy and losses with the help of different matrices such as average forgetting score, average accuracy and parameters is given in table.5. This evaluation incorporates diverse metrics, offering a comprehensive understanding of the proposed architecture.

Table 5. Is summary of how different neural network architectures perform on various image classification tasks. The architecture column shows the name of the architectures, which indicates the type, depth, and width of the model. Zagoruyko et al. [45] proposed a model which represents a wide residual network with a depth of 10 and a widening factor of 2, is a notable convolutional neural network design renowned for its effectiveness in image classification tasks. This architecture extends the foundational ideas from residual networks (ResNets). The central concept behind WRN involves enhancing model capacity by widening the network while maintaining a relatively shallow depth, thus mitigating overfitting concerns and improving overall generalization performance. WRN-10-10, denoting a wide residual network with a depth of 10 and a widening factor of 10, stands out as a convolutional neural network design celebrated for its prowess in handling image classification tasks. The core principle underlying WRN-10-10 involves augmenting the model's capacity by broadening the network while maintaining a relatively shallow depth, thereby addressing concerns related to overfitting. He et al. proposed the ResNet-34, ResNet-50, and ResNet-101 vary in their depth, comprising 34, 50, and 101 layers, respectively. The numerical values in their names correspond to the total count of convolutional layers within the architecture. These ResNet variants employ residual blocks, which are pivotal for training extremely deep neural networks by mitigating the issue of vanishing gradients. Within a ResNet, each residual block consists of multiple convolutional layers, followed by batch normalization and ReLU given by Nair et al. [46] activation functions. Notably, these blocks incorporate shortcut connections, also known as skip connections, allowing the gradient to propagate more smoothly during training. This mechanism effectively tackles the challenge of vanishing gradients and facilitates the training of deeper networks. ResNet-34 exhibits a

simpler structure compared to ResNet-50 and ResNet-101 due to its fewer layers. On the other hand, ResNet-50 and ResNet-101, with their deeper architectures, possess the capability to capture intricate features from input data. However, these deeper variants demand more computational resources for both training and inference tasks. For instance, GemResNet-18 means a Generalized-Memory ResNet model with 18 layers, and WRN-10-10 means a Wide ResNet model with 10 layers and a width factor of 10. The average forgetting score is a measure of how much the model's performance on a previous task deteriorates after learning a new task. A lower forgetting score means that the model is more capable of preserving its knowledge from previous tasks. For instance, VResNet4 has the lowest forgetting score of 11, which means that on average, its accuracy on a previous task reduces by 11% after learning all the tasks. The Average Accuracy column shows the average accuracy of each architecture, which is the percentage of correctly classified images across all the tasks.

Table 5. Effect of parameters on forgetting and accuracy

Previous Presented Method	Architecture	Average Forgetting score	Average Accuracy	Parameters/1000,000
Chow et al.	GemResNet-18	15	64.7	10.9
	WRN-10-2	33	47.2	0.3
	WRN-10-10	28	53.8	7.7
Zagoruyko et al.	WRN-16-2	39	41.3	0.7
	WRN-16-10	34	49.9	17.5
	WRN-28-2	37	44.2	6
	WRN-28-10	33	47.1	36.6
	ResNet-34	54	47.3	21.5
He et al.	ResNet-50	55	57.2	23.8
	ResNet-101	54	58.1	42.7
	VResNet1	18	62.52	12.9
Proposed Architecture	**VResNet2**	**14**	**68.9**	**43.1**
	VResNet3	**13**	**68.58**	**17.3**
	VResNet4	**11**	**69.78**	**32.99**

A higher accuracy means that the model is more accurate for the given tasks. VResNet4 has the highest accuracy of 69.78%, which means that it correctly classifies about 70% of the images on average. Parameters/1000,000 is a column that shows the number of parameters in each architecture divided by one million, which is an indicator of the model's size and complexity. A higher number of parameters means that the model has more weights and biases to train, which may increase its expressive power but also its computational cost and risk of overfitting. For instance, ResNet-101 has the highest number of parameters at 42.7 million, which means that it is a very large and complex model. The study shows that VResNet4 is the best-performing architecture among the ones listed, as it has the lowest forgetting score, the highest accuracy, and a moderate number of parameters. However, this does not mean that VResNet4 is always the best choice for any image classification task, as different tasks have different requirements and trade-offs. For example, if we want to save memory and computational resources, we may prefer a smaller architecture like WRN-10-2, which has only 0.3 million parameters.

In Fig. 11. From the graph, we have analyze that VResnet1 has a low accuracy of about 45% on the first task, but it improves gradually as it learns from more tasks. It shows a positive trend in performance and attains an accuracy of around 72% by task7.

However, its performance becomes unstable and sometimes drops after task 7. The model has significant variations in performance, especially after task 12.VResNet2 has a high accuracy of about 57% on the first task and performs consistently better than VResNet1. It achieves an accuracy of around 82% by task 5. VResNet2 maintains its high performance and has fewer variations than VResnet1.VResNet3 has a similar accuracy to VResNet2 of about 59% on the first task, and its performance is comparable to VResNet2 in the early tasks. It reaches an accuracy of around 83% by task 5. However, like VResnet1, it also has unstable performance after task 10. VResNet4 has an accuracy of about 57% on the first task and keeps a high and stable performance throughout the tasks. It attains an accuracy of around 80% by task 6 and maintains a relatively constant accuracy in later tasks. VResNet4 is the most stable model among the four. In a continual learning scenario, where models learn from multiple tasks sequentially, maintaining stability and preventing catastrophic forgetting are important. VResNet4 shows better stability, while VResNet2 also performs well. VResnet1 and VResNet3 may require techniques to reduce performance variations and forgetting of previous tasks. It's important to consider the characteristics of these models when choosing the appropriate model for a specific application in a continual learning setting. Additionally, techniques such as lifelong learning, knowledge distillation, and architectural modifications can be explored to improve the performance and stability of these models.

Fig. 12 shows the comparative analysis of different models on continuous learning tasks. VResnet1 has a relatively high initial loss on the first task, showing that the model's performance is not

Optimal at the beginning. The model shows improvement as it learns from more tasks and reduces its losses gradually. It reaches a relatively low and stable loss by task 8, showing that the model has learned effectively from the tasks. There are some variations in loss after task 8, but they do not indicate a significant increase in forgetting. VResNet2 has a lower initial loss on the first task than VResNet1, showing better initialization. The model reduces its loss consistently with each new task and shows a downward trend, indicating continuous learning and improvement. After task 10, the loss keeps decreasing, showing that the model learns effectively from new tasks. VResNet2 has minimal loss variations, showing that it retains knowledge from previous tasks well. VResNet3 has a relatively high initial loss on the first task, similar to VResNet1, showing that the initial performance is suboptimal. The model reduces its loss gradually and achieves a lower loss by task 5, showing improved task-specific performance. However, from task 10 onwards, there is a significant increase in the loss,

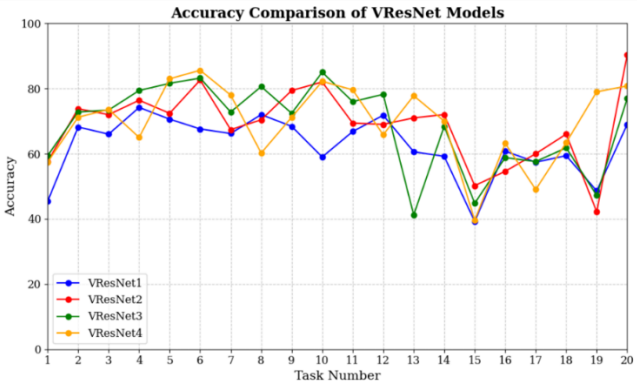

Fig.11. Accuracy of all architecture over Task Numbers

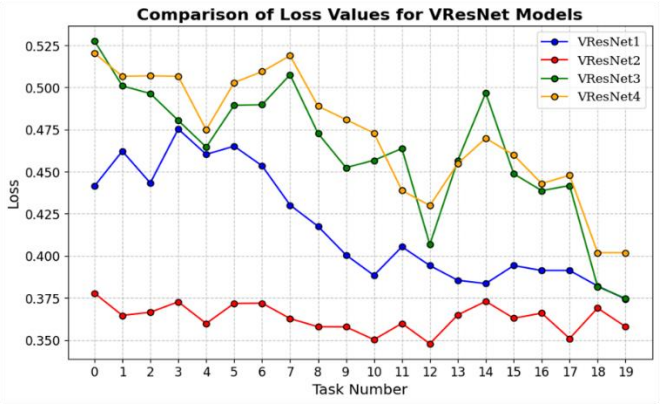

Fig.12. Loss of all architecture over Task Numbers

Showing potential catastrophic forgetting of previous tasks. The variation in loss after task 10 shows that the model has difficulty retaining knowledge from earlier tasks. VResNet4 starts with a moderately low initial loss on the first task, showing a good starting point. The model keeps a consistently low loss throughout the tasks, showing its ability to learn from new tasks without significant forgetting. VResNet4 is relatively stable and does not show significant variations in loss, showing good knowledge retention. VResNet4 shows superior performance among the models, with a consistently low loss throughout tasks, showing strong continual learning capabilities.

VResNet2 also shows good performance, with a gradually decreasing loss and minimal fluctuations, showing effective knowledge retention. VResNet1, although showing improvement over tasks, has some fluctuations in loss, showing occasional forgetting, but it stabilizes at a reasonably low loss. VResNet3 starts with high loss and shows significant forgetting after task 10, with noticeable fluctuations in loss. In continuous learning scenarios, it is important to consider models that can effectively manage knowledge retention and minimize catastrophic forgetting. VResNet4 and VResNet2 show promising characteristics in this regard. Further research and techniques can be explored to address the challenges faced by VResNet1 and VResNet3, such as regularization methods and architectural modifications to mitigate forgetting and improve stability in learning from sequential tasks

6. Conclusion

This research explores the impact of different architectural attributes on continual learning, including shape, size, and parameters. Additionally, it proposes several variants of ResNet, a widely used model in computer vision. Our study reveals that the architecture's shape, width, depth, and utilization of max-pooling significantly influence its performance and susceptibility to forgetting in continual learning scenarios. Notably, we observe that enhancing the width of the architecture and employing appropriate shapes with a balanced depth-to-width ratio ratio leads to improved outcomes with reduced forgetting. Moreover, our findings indicate that augmenting the architecture's depth does not markedly alter its loss value across diverse tasks. This study sheds light on the intricate relationship between architectural design and continual learning performance, providing insights for optimizing model configurations to mitigate forgetting and enhance overall performance in AI systems. This study has some limitations that can be addressed in future research. First, the research only used one dataset and one type of model for our experiments. Extending the research to other datasets and models, such as transformers, would be interesting, which have shown promising results in various domains. Second, we used fixed architectures for each task without adapting them to the task-specific requirements.

Conflicts of interest

The authors declare no conflicts of interest.

References

[1] Yin, Q.Y., Yang, J., Huang, K.Q., et al., "AI in Human-computer Gaming: Techniques, Challenges and Opportunities," Mach. Intell. Res., vol. 20, pp. 299–317, 2023. [Online]. Available: https://doi.org/10.1007/s11633-022-1384-6

[2] O. Vinyals et al., "Grandmaster level in Starcraft II using multi-agent reinforcement learning," Nature, vol. 575, no. 7782, pp. 350–354, 2019. [Online]. Available: https://doi.org/10.1038/s41586-019-1724-z

[3] S. Thrun, "Lifelong learning algorithms," in Learning to Learn, pp. 181–209, Springer, 1998.

[4] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," Psychology of Learning and Motivation, vol. 24, pp. 109–165, 1989.

[5] J. Yoon et al., "Lifelong learning with dynamically expandable networks," in Sixth International Conference on Learning Representations (ICLR), 2018.

[6] M. Delange et al., "A continual learning survey: Defying forgetting in classification tasks," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021. [Online]. Available: https://doi.org/10.1109/TPAMI.2021.3057446

[7] G. M. Van de Ven and A. S. Tolias, "Three scenarios for continual learning," arXiv preprint arXiv:1904.07734, 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1708.01547

[8] K. He et al., "Deep residual learning for image recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778, 2016. [Online]. Available: https://doi.org/10.48550/arXiv.1512.03385

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, no. 6, 2017. doi: 10.1145/3065386

[10] C. Badue et al., "Self-driving cars: A survey," CoRR, abs/1901.04407, 2019.

[11] D. Silver et al., "Mastering the game of go with deep neural networks and tree search," Nature, vol. 529, no. 7587, pp. 484–489, 2016.

[12] Y. Li et al., "Provable and Efficient Continual Representation Learning," arXiv:2203.02026v2 [cs.LG], 2022.

[13] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7765–7773, 2018. doi: 10.48550/arXiv.1711.05769

[14] J. T. Wixted, et al., The psychology and neuroscience of forgetting, Annual review of psychology 55 (1) (2004) 235–269. [Online]. Available: https://doi.org/10.1146/annurev.psych.55.090902.141555

[15] M. Ye, X. Zhang, P. C. Yuen, S.-F. Chang, Unsupervised embedding learning via invariant and spreading instance feature, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 6210–6219.

[16] D. Lopez-Paz and M. Ranzato, "Gradient Episodic Memory for Continual Learning," in Advances in Neural Information Processing Systems, vol. 30, 2017, pp. 6467-6476.

[17] D. Yin et al., "Optimization and generalization of regularization-based continual learning: a loss approximation viewpoint," arXiv preprint arXiv:2006.10974, 2020.

[18] P. Buzzega, M. Boschini, A. Porrello, and S. Calderara, "Rethinking experience replay: a bag of tricks for continual learning," in Proc. 25th Int. Conf. Pattern Recognit. (ICPR), 2021, pp. 2180–2187.

[19] J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," Proc. Natl. Acad. Sci. U.S.A., vol. 114, no. 13, pp. 3521–3526, 2017.

[20] C. Fernando et al., "Pathnet: Evolution channels gradient descent in super neural networks," arXiv preprint arXiv:1701.08734, 2017.

[21] M. Wortsman et al., "Supermasks in superposition," arXiv preprint arXiv:2006.14769, 2020.

[22] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, C. H. Lampert, icarl: Incremental classifier and representation learning, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, 2017, pp. 5533–5542.

[23] S. Hou, X. Pan, C. C. Loy, Z. Wang, D. Lin, Learning a unified classifier incrementally via rebalancing, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 831–839. doi: 10.48550/arXiv.1812.00420

[24] A. Chaudhry, R. Marc'Aurelio, M. Rohrbach, M. Elhoseiny, Efficient lifelong learning with agem, in: 7th International Conference on Learning Representations, ICLR 2019, International Conference on Learning Representations, ICLR, 2019.

[25] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Y. Fu, Large scale incremental learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 374–382.

[26] Q. Pham, D. Sahoo, C. Liu, S. C. Hoi, Bilevel continual learning, arXiv preprint arXiv:2007.15553 (2020).

[27] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE transactions on pattern analysis and machine intelligence 35 (8) (2013) 1798–1828. doi: 10.1109/TPAMI.2013.50. doi: 10.1109/TPAMI.2013.50

[28] D. P. Kingma, M. Welling, Auto-encoding variational bayes, arXiv preprint arXiv:1312.6114 (2013).

[29] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Communications of the ACM 63 (11) (2020) 139– 144.

[30] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Y. Fu, Large scale incremental learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 374–382. doi: 10.1109/TPAMI.2013.50

[31] H. Shin, J. K. Lee, J. Kim, J. Kim, Continual learning with deep generative replay, arXiv preprint arXiv:1705.08690 (2017).

[32] O. Ostapenko, M. Puscas, T. Klein, P. Jahnichen, M. Nabi, Learning to remember: A synaptic plasticity driven framework for continual learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 11321–11329. doi: 10.1109/TPAMI.2013.50

[33] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in Proc. 34th Int. Conf. Mach. Learn., vol. 70, 2017, pp. 3987–3995. doi: 10.1109/TPAMI.2013.50

[34] A. Chaudhry et al., "On tiny episodic memories in continual learning," arXiv preprint arXiv:1902.10486, 2019.

[35] K. Cho, T. Raiko, and A. Ilin, "Enhanced gradient and adaptive learning rate for training restricted boltzmann machines," in Proc. 28th Int. Conf. Mach. Learn. (ICML), 2011, pp. 105–112. doi: 10.1109/TPAMI.2013.50

[36] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in Int. Conf. Mach. Learn., 2018, pp. 4548–4557.

[37] S. Lee, S. Behpour, and E. Eaton, "Sharing less is more: Lifelong learning in deep networks with selective layer transfer," in Proc. 38th Int. Conf. Mach. Learn., vol. 139, 2021, pp. 6065–6075.

[38] A. Mallya, D. Davis, and S. Lazebnik, "Piggyback: Adapting a single network to multiple tasks by learning to mask weights," in Proc. Eur. Conf. Comput. Vis. (ECCV), 2018, pp. 67–82. doi: 10.1109/TPAMI.2013.50

[39] X. Li et al., "Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting," in Proc. 36th Int. Conf. Mach. Learn. (ICML), vol. 97, 2019, pp. 3925–3934.

[40] B. H. Y. Chow and C. C. Reyes-Aldasoro, "Automatic Gemstone Classification Using Computer Vision," Minerals, vol. 12, 2022, Art. no. 60. doi: 10.3390/min12010060.

[41] J. S. Smith et al., "A Closer Look at Rehearsal-Free Continual Learning," in 2023 IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR) Workshop on Continual Learning. Comput. Vis. (CLVision 2023).

[42] B. Zhou et al., "Learning deep features for discriminative localization," in 2016 IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 2921–2929. doi: 10.1109/TPAMI.2013.50

[43] M. Lin, Q. Chen and S. Yan, “Network In Network,” 2014 International Conference on Learning Representations (ICLR), Banff, AB, Canada, 2014.

[44] P. Sermanet, S. Chintala and Y. LeCun, “Convolutional Neural Networks Applied to House Numbers Digit Classification,” 2012 21st International Conference on Pattern Recognition (ICPR), Tsukuba, Japan, 2012, pp. 3288-3291. doi:10.3390/min12010060

[45] S. Zagoruyko and N. Komodakis, "Wide Residual Networks," in Proceedings of the British Machine Vision Conference (BMVC), 2016, doi: 10.5244/C.30.84

[46] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML), Haifa, Israel, 2010, pp. 807-814.