Lifelong Learning using Neural Networks

by AbdulRahman Tiba

Loughborough University

22COP328: Data Science Project

Supervised by Dr Shirin Dora

Submitted on 18th of September, 2023

Acknowledgements

First and foremost, my immense gratitude and acknowledgement goes to my dissertation supervisor, Dr Shirin Dora. His extensive knowledge, astute guidance, illuminating insights, invaluable feedback and persistent support were instrumental in shaping this project. It is impractical to articulate just how focal he has been in guiding this project.

I would also like to extend my sincere appreciation to all of my lecturers and teaching assistants who contributed to my academic development over the past year. In particular, I would like to thank Dr Firat Batmaz for imparting his expertise in Python programming, which proved invaluable for my research methodology. My gratitude also goes to Dr Hui Fang for sharing his extensive knowledge in artificial intelligence and applied machine learning, which formed the backbone of this project.

To my family, I owe a lifetime of gratitude and acknowledgement. To my parents, whose love knows no bounds and whose guidance, wisdom, and countless sacrifices have been my unwavering pillars of strength and support. And to my brother, my superlative source of strength and lifelong confidant.

I owe immeasurable gratitude to my UWE Bristol Business and Management undergraduate programme leader, mentor and friend, Dr Neil Sutherland. He has been an unyielding source of support, guiding me through the intricacies of academia and life's challenging moments.

Additionally, I would like to express my profound gratitude to Dr Stephen Bennington, my CEO during my tenure at Q5D Technology. His inspiration and mentorship were pivotal in guiding me to pursue this degree. His inexorable belief in constant learning and growth resonates with me, and I am immensely grateful for the influence he has had on my journey.

To all my treasured friends, your presence has been a sanctuary, allowing me to find balance and soundness amidst the rigours of this research journey. Our heartfelt conversations and shared moments have been the wellspring of my resilience.

I was incredibly fortunate to go through this journey alongside an amazing group of coursemates. Our solidarity in working through difficult coursework and moments of solace has made this journey all the more rewarding. The fellowship, mutual encouragement, and collective passion for our field are things I will always cherish.

I must also acknowledge and thank the sweet lady from the Pilkington Library cafeteria. Who fuelled my innumerable study hours in the library with endless cups of coffee, always served with a smile. Her small acts of service and kindness were often the boost I needed to tackle another chapter or to read another paper.

To everyone mentioned here and those not mentioned but who contributed in ways, principal or minor- my sincere appreciation. I am profoundly humbled and eternally grateful.

Lifelong Learning using Neural Networks

AbdulRahman Tiba

Abstract

Continual learning remains an enduring challenge in artificial intelligence as models struggle to assimilate new information without forgetting previously acquired knowledge, an issue referred to as catastrophic forgetting. Contemporary deep neural networks exhibit heightened susceptibility, given their inherent rigidity. This research explores the emerging concept of supermasks, identified as high-performing sparse sub-networks within larger over-parameterised networks, as a potential remedy. The chief aim is to comprehensively investigate the extent of representational overlap between supermasks pertaining to distinct tasks and how such convergence can be strategically utilised to enhance continual learning efficiency without compromising performance. Three key objectives motivate this study: quantifying the overlap between supermasks across varied MNIST dataset tasks, pioneering a novel learning methodology to exploit similarity for faster learning, and iteratively refining the proposed approach through rigorous experimentation and analyses. The extent of overlap is empirically measured using the Jaccard similarity index and Cosine similarity. A nuanced framework integrates estimated similarity through a weighted combination of supermasks. Comparative evaluations against a classic technique across accuracy, training time, loss and advanced metrics like precision, recall and F1-score provide critical insights. The results reveal extensive yet nuanced overlap, quantifying and corroborating representational sharing. The proposed integration of similarity demonstrates substantial improvements in training efficiency. However, performance advantages are contingent on precise methodologies and datasets involved. While clearly promising, explicitly modelling overlap does not confer universal benefits. Supermasks overlap exhibits significant yet contextually-dependent potential to accelerate learning without detriment to performance. The exposed limitations provide constructive directions. Future enhancements should focus on architectural and methodological refinements, statistical rigour, transformational breadth, and meticulous, hierarchical integration of similarities. Fulfilling the latent promise warrants persistent, iterative refinement.

Table of Contents

List of Figures

List of Tables

"The supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience."

— Albert Einstein, 1933

Chapter 1: Introduction

1.1 Background

The notion of authenticating knowledge acquisition, as derived from incessant experiences, has invariably remained at the nucleus of the illustrious journey of Artificial Intelligence (AI) and Machine Learning (ML) disciplines, drawing rich analogies from the ontological evidence presented by biological entities (Chen and Liu, 2018). Historically, due to a confluence of the sparsity of digital information and the relatively primordial complexity of the initial quandaries broached, the conception of most machine learning paradigms bore the hallmarks of staticity; a temporal stasis where continual adaptation was evidently inessential.

With the advent of the 21st century and the ensuing Big Data revolution, the viability of this perspective appears to be diminishing for a burgeoning array of applications engendered by the precipitous escalation in data volume, variability, velocity, and veracity, as elucidated by (Laney, 2001). However, during the inaugural decade, the research community was preoccupied with addressing pressing issues such as numerical optimisation and ad-hoc feature selection, which have been instrumental in enhancing the capabilities of AI learning systems in multifarious complex domains characterised by high-dimensional data, including computer vision, speech recognition, and others (Goodfellow, Bengio and Courville, 2016).

The metamorphic emergence of Deep Learning (DL), as delineated by (LeCun, Bengio, and Hinton, 2015), and especially the seminal research conducted by (Krizhevsky, Sutskever, and Hinton, 2012) dramatically expanded the capacity to acquire highly nuanced and adaptable representations directly from raw data, paving the way for a broader repertoire of applications whose complexity was hitherto inconceivable. Moreover, the encapsulation and isolation of learning challenges into discrete domains or tasks have become progressively intractable. Novel learning algorithms, analogous to their biological counterparts, would ideally necessitate access to superabundant amounts of high-dimensional, multi-domain, streaming data emanating from intricate and continuously evolving environments in order to augment their intelligence and effectively adapt to developing circumstances over time. Such is the cornerstone of achieving true algorithmic intelligence and temporally robust adaptability (Kaiser et al., 2017).

The subject of Continual Learning (CL) has garnered increasing attention (Parisi et al., 2019), catalysing this renewed and exponential interest. Unfortunately, contemporary deep learning approaches encounter several obstacles in gleaning knowledge from a ceaseless influx of data, as elucidated by (Goodfellow et al., 2013). Typically, predictive models are exclusively trained on predefined and representative datasets, which impedes their capacity to effectively extend or refine their inherent plasticity over time. Moreover, when a model is sequentially trained on multiple tasks with divergent data distributions, it may exhibit degradation in performance on previously learned tasks, an issue referred to as catastrophic forgetting.

To mitigate catastrophic forgetting, numerous strategies have been proposed, including Elastic Weight Consolidation (EWC) introduced by (Kirkpatrick et al., 2017). These impose modulatory constraints to regulate parameter changes during new task learning. However, the issue of forgetting remains profound as the number of tasks accretes, even when employing these techniques. Memory-based approaches, which store pertinent information about learned tasks to maintain performance, have also been applied to address forgetting. Furthermore, various methods to mitigate forgetting in multi-domain learning have been proposed. For instance, (Mallya, Davis and Lazebnik, 2018) and (Mancini et al., 2018) introduced approaches that alleviate forgetting by introducing a minimal number of additional parameters while keeping prior estimated parameters static. However, these models are heavily reliant on a robust backbone network, and the extent of transferable knowledge is largely limited.

The majority of existing CL methodologies integrate network architecture and parameter estimation, typically employing an invariant model structure across tasks. Conversely, limited techniques have been developed to mitigate parameter interference and catastrophic forgetting by drawing inspiration from the Lottery Ticket Hypothesis proposed by (Frankle and Carbin, 2018) and the notion of Supermasks introduced by (Zhou et al., 2019). By harnessing the capabilities of sparse subnetworks, these approaches aim to unlock the expressive potential of neural networks while tackling the aforementioned challenges. (Zhou et al., 2019) observed that within randomly initialised neural networks, specific subnetworks referred to as supermasks exhibit superior performance, a remarkable finding given the vast number of possible sparse subnetworks. A supermask constitutes a sparsely binary mask that selectively preserves or prunes individual connections within a predefined randomly initialised network, with the objective of identifying a subnetwork that demonstrates admirable performance on a given task without catastrophic forgetting. This involves a multitask neural network wherein each task has an associated mask. By harnessing the latent potential of sparse subnetworks, these approaches unravel the heretofore unexplored domains of artificial neural network plasticity and expressivity.

The prime objective of this study is to conduct an exhaustive analysis of the literature on continual learning and supermasks, introducing a novel strategy to address catastrophic forgetting using supermasks. The degree of overlap among masks for different tasks within the 'What's Hidden in a Randomly Weighted Neural Network' and 'Supermasks in Superposition' framework (Ramanujan et al., 2020; Wortsman et al., 2020) will be measured and analysed to ultimately decipher if such overlaps can be leveraged to catalyse a more efficient and efficacious continual learning trajectory. The term 'overlapping' in this context implies that certain neurons are shared among multiple tasks. A systematic methodology will be developed to improve model performance across multiple tasks over time. Comprehensive benchmarking on challenging variants of the MNIST dataset will be performed to compare the proposed model to a state-of-the-art technique.

1.2 Motivation

The motivating factors for this research can be delineated as follows:

- As the data landscape grows in complexity, traditional ML models face challenges adapting to changing data streams. While static paradigms struggle in real-time scenarios, lifelong learning models, in contrast, offer a robust solution by promising techniques that assimilate knowledge continuously, ensuring sustained performance amidst evolving data and tasks.

- A predominant issue facing neural networks in a lifelong learning paradigm is catastrophic forgetting, wherein the assimilation of new data can lead to the overwriting of previously learned patterns. This research aims to delve into mechanisms and neural architectures that can mitigate this challenge, allowing for the seamless incorporation of new information without jeopardising previously acquired knowledge.

- The concept of supermasks proposes an intriguing remedy to the forgetting conundrum. Their ability to identify high-efficacy sparse subnetworks within deep artificial neural networks, coupled with their proclivity to enhance task-specific proficiency, underscores their seminal relevance.

- Analysing and exploring the extent of convergence between task-specific masks within the 'What's Hidden in a Randomly Weighted Neural Network' and 'Supermasks in Superposition' framework presents a valuable opportunity to harness

shared knowledge, potentially boosting the performance of ML systems on complex tasks by effectively assimilating and disseminating collective knowledge. The strategic leverage of this collective acumen can potentially catapult ML systems to hitherto unprecedented degrees of proficiency in multifaceted tasks.

1.3 Research Objectives

The structure of our research objectives can be architecturally delineated as follows:

- Perform an exhaustive survey and critical appraisal of current academic literature on catastrophic forgetting in neural networks for continual learning, analysing the various techniques proposed to address this challenge, with a focus on the notion of supermasks.

- Determine the extent of mask overlap across different tasks within the ‘What’s Hidden in a Randomly Weighted Neural Network’ and ‘Supermasks in Superposition’ framework.

- Introduce and refine a novel methodology anchored in the aforementioned framework with the intent to gauge if such overlaps can amplify the learning efficiency of subsequent tasks, potentially delineating a roadmap for more efficient continual learning.

- Empirically vet our methodology through rigorous experimentation on multiple variants of a benchmark dataset and against an established baseline model to discern its relative merit. This facet of our research is instrumental in equipping our constructs with empirical veracity and operational credibility.

1.4 Outline

The subsequent chapters meticulously unravel our research study:

- **Chapter 2** offers an incisive survey of the literature, placing a pronounced emphasis on the intricacies of continual learning and the strategies, both conventional and innovative, developed and deployed to counteract it. Moreover, it delves into the bedrock principles of the supermasks paradigm, elucidating its operational mechanics, empirical ascendency, and innate potential.

- **Chapter 3** introduces and delineates the architectural contours of our methodology based on the aforementioned substructure, offering a lucid elucidation of its theoretical underpinnings and operational mechanics.

- **Chapter 4** chronicles our empirical experiments, explicating the experimental design, datasets employed, and the ensuing results. A rigorous comparative analysis against an established baseline model bestows empirical legitimacy on our constructs.

- **Chapter 5** concludes the narrative, reflecting upon the gleaned insights, potential implications, research limitations and the prospects for future exploration in this domain.

Chapter 2: Literature Review

2.1 Overview

Real-world computational systems are subjected to continuous data streams and must cultivate and retain multiple tasks from dynamic data distributions. An autonomous agent's exchange with its environment mandates continuous learning based on distinct experiences. This procedure entails gradually acquiring knowledge, which is constantly refined and improved. In addition, the agent should be able to transmit this acquired knowledge over extended periods of time. Given that data is sequentially streamed from various distributions, it is crucial for the continual learning agent to effectively transfer knowledge between tasks, all the while maintaining performance on the observed data.

(McCloskey, 1989) addressed catastrophic network interference. Initially, several properties of distributed representations were created by enforcing learning algorithms, which are considered desirable from the perspective of modelling human cognition. The aforementioned attributes encompass content-addressable memory and a phenomenon known as automatic generalisation, wherein a network that has undergone training on a specific collection of items exhibits appropriate responses to untrained items falling within the same domain. When networks are trained sequentially, new learning may conflict catastrophically with old learning. The investigation of the underlying factors contributing to interference suggests that the occurrence of interference is inevitable whenever new learning has the potential to modify the weights utilised in representing previously acquired knowledge. In contrast, the simulation findings only show that interference is catastrophic in specific networks. In the context of lifelong learning, the susceptibility of computational models to catastrophic forgetting or catastrophic interference is a significant obstacle.

According to (McClelland, 1995), there exists a phenomenon wherein the process of training a model with novel data results in an interruption or deletion of previously acquired knowledge. Since the evolution of neural networks, researchers have extensively investigated this phenomenon and speculated that the sharing of parameters, which facilitates the generalisation ability of neural networks based on observed data, is the underlying cause of this issue.

2.2 Foundations and Evolution of Artificial Intelligence

Artificial Intelligence (AI) convolutes fields like computer science, mathematics, psychology, neuroscience, cognitive science, and more. From its inception, AI aimed to replicate human thought, reasoning, and decision-making. As outlined by (Russell and Norvig, 2021), the goal is for the machines to perform cognitive functions akin to or exceeding humans with minimal oversight.

Initial AI efforts leaned on knowledge bases, using hard-coded statements for decision-making. However, these systems had constraints due to humans' inability to express every nuance of complex tasks (Brynjolfsson and McAfee, 2017). This prompted the rise of Machine Learning (ML), as detailed by (Jordan and Mitchell, 2015). ML algorithms refine performance using repeated data, identifying patterns not directly programmed (Bishop, 2006). ML's adaptability, especially in tasks like classification and regression, marks a crucial step in AI development.

Delving deeper, Artificial Neural Networks (ANNs) stand out. Mirroring biological systems, ANNs consist of linked artificial neurons—these connections channel signals modulated by weights that shift during learning. Signal processing is determined by an activation function, and the network might have multiple layers (Bishop 2006; Goodfellow et al. 2016).

From ANNs, Deep Neural Networks (DNNs) evolved. Deep learning, a branch of ML, enhances neural networks by adding multiple layers between input and output—hence the term 'deep' (LeCun et al. 2015). This depth enables hierarchical learning, with layers capturing varying abstraction degrees. The transition from basic neural networks to deep models has spurred progress in areas like computer vision and language processing. This growth delineates between 'shallow' ML and deep learning, as illustrated in Figure 1. While basic ANNs are transparent or 'white boxes', DNNs' complexities often render them 'black boxes', with opaque decision-making processes.

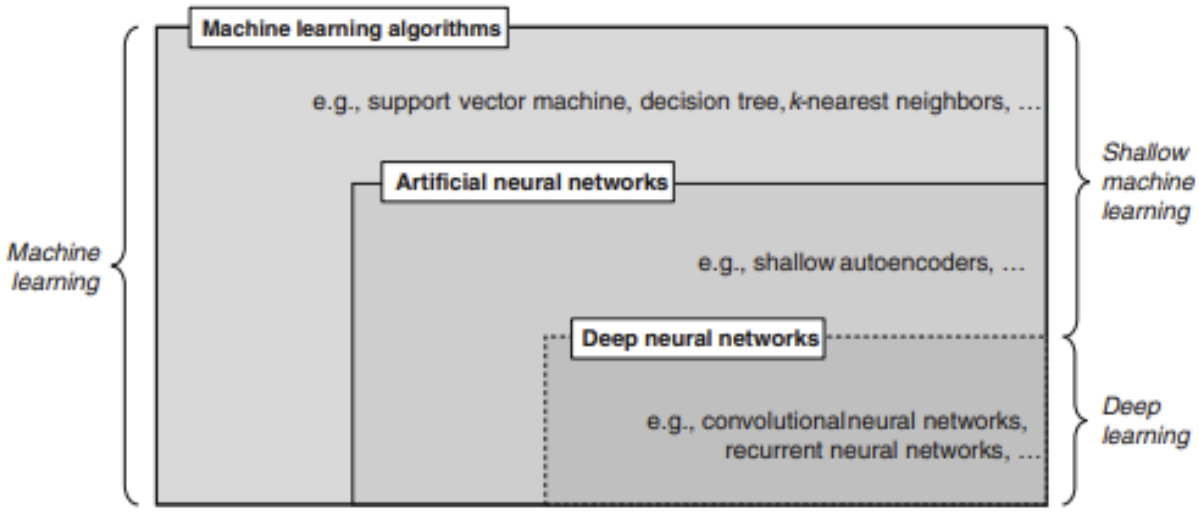

Figure 1: Venn diagram of ML concepts and classes (Janiesch, Zschech and Heinrich, 2021)

2.3 Foundations of Neural Networks

Neural networks, in their essence, are computational models that draw emulation from the intricate neural structures found within biological entities capable of cognition and sentient behaviours. As they replicate certain key functionalities of the brain's neurons, they have proven to be paramount in advancing the domain of AI as they ushered a departure from explicitly programmed algorithms to systems that can intricately comprehend and learn from data.

Drawing deeply from the neuronal structures of sentient organisms, these networks consist of layers of interconnected nodes or 'neurons'. Each connection or 'synapse' possesses a weight that adjusts during the learning process, moulding the network's predictive or representative abilities. The remarkable ability of these networks to assimilate and generalise knowledge is accredited to a rigorous pedagogical process, where the network is meticulously trained with vast quantities of data, honing its predictive capabilities over time.

At the core of neural network operation lies the activation function, which introduces non-linearities, enabling the network to capture intricate patterns and relationships in data. The training phase of these networks involves optimising a cost or loss function, which quantifies the deviation between the network's predictions and the actual values, using optimisation algorithms like gradient descent.

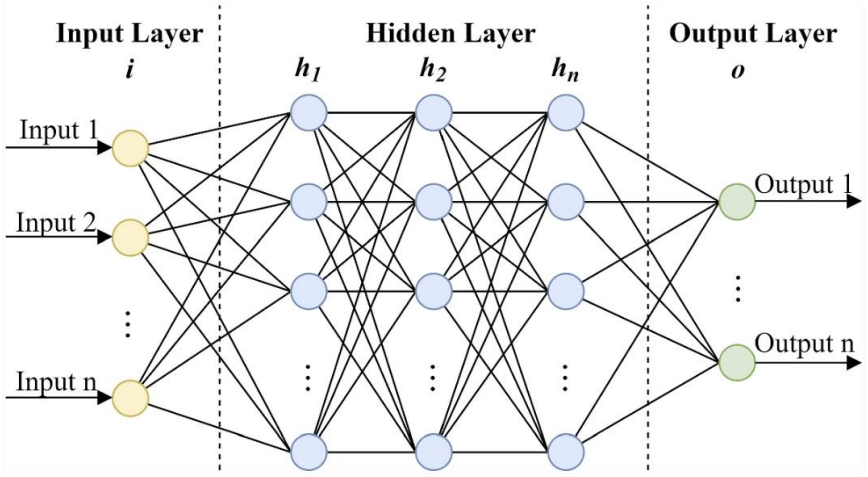

Figure 2: Illustration of a general neural network architecture (Balakrishnan et al., 2021)

2.4 Architectures and Evolution of Neural Networks

Feedforward Neural Networks (FNN) stand as perhaps the most foundational form of neural network architectures. Elucidated by (Sazli, 2006), these networks represent a linear progression, where data flows directly from the input layer to the output layer, devoid of any cyclic or recurrent paths. Their simplicity has been a cornerstone in the nascent stages of neural network applications, laying the groundwork for more intricate architectures.

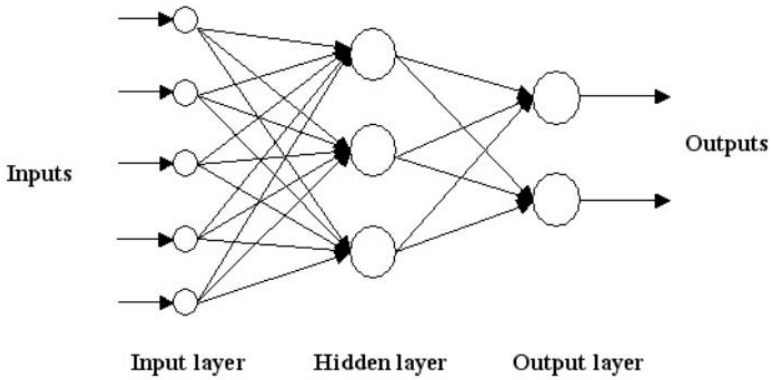

Figure 3: Feed Forward Neural Network architecture (Sazli, 2006)

Furthermore, Recurrent Neural Networks (RNN) is a marked evolution from FNNs, designed explicitly to cater to sequences and temporal data. As detailed by (Sherstinsky, 2020), these networks possess the ability to retain past information through internal loops, making them exceptionally suitable for tasks like speech recognition and time-series analysis.

However, their prowess is not without challenges; they often grapple with issues like vanishing and exploding gradients when dealing with long sequences.

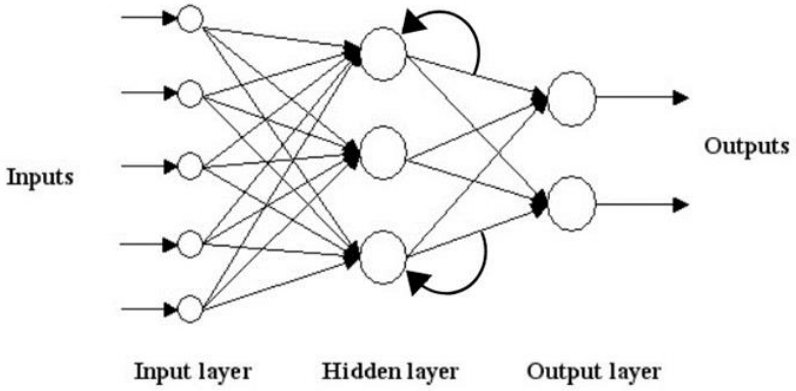

Figure 4: Recurrent Neural Network architecture — inspired by (Balakrishnan et al., 2021; Sherstinsky, 2020)

An answer to the challenges faced by traditional RNNs can be delineated in the form of Long Short-Term Memory (LSTM) Networks. LSTMs, initially elucidated by (Hochreiter and Schmidhuber, 1997), were ingeniously developed to circumvent the problems of vanishing and exploding gradients. The introduction of gating mechanisms within the architecture allows them to regulate the flow of information and thus capture long-term dependencies in sequence data with remarkable adeptness.

In the realm of spatial data, particularly images and videos, Convolutional Neural Networks (CNN) have asserted dominance. With their foundational principles delineated by (Lecun et al., 1998), CNNs utilise convolutional layers to detect local patterns in data. The hierarchical nature of these layers enables them to capture increasingly abstract features, making them indispensable for image classification, detection, and even generative tasks.

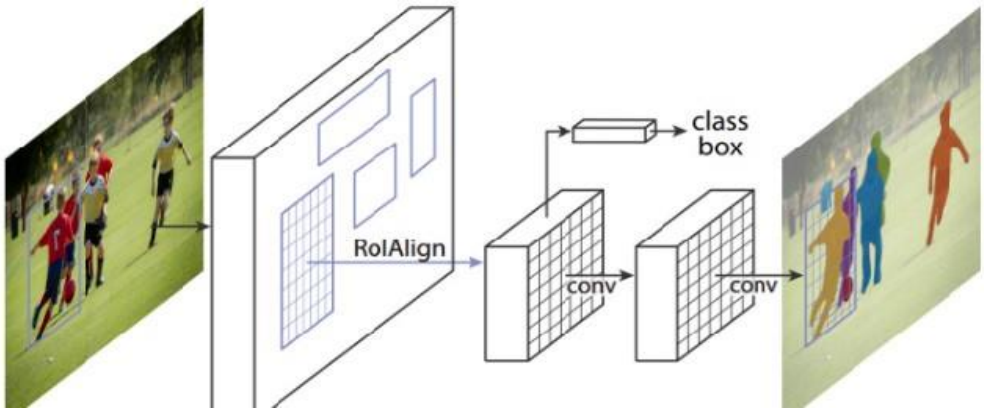

Figure 5: Illustration of Convolutional Neural Network architecture (He et al., 2017)

The concept of Autoencoders epitomises unsupervised learning within neural network architectures. As propounded by (Bank, Koenigstein and Giryes, 2021), an autoencoder consists of an encoder segment, which compresses input data, and a decoder segment, which strives to reconstruct the original input. Beyond their initial applications in data compression, they have shown promise in anomaly detection and even generative tasks.

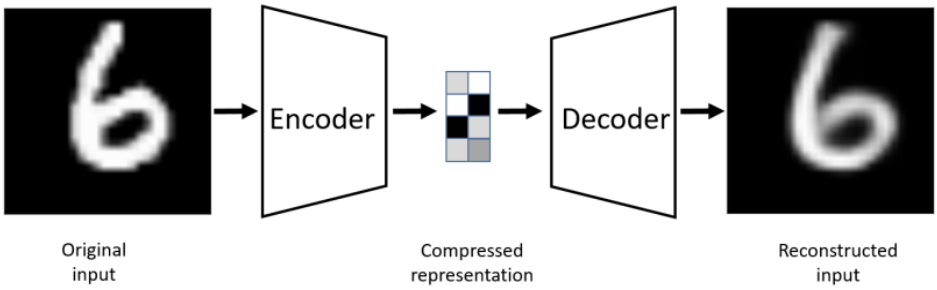

Figure 6: Illustration of Autoencoder Neural Network (Bank, Koenigstein and Giryes, 2021)

A more recent development in unsupervised learning is the Generative Adversarial Network (GAN). Introduced by (Goodfellow et al., 2014), GANs consist of two neural networks—a generator and a discriminator—that engage in a form of game theory. While the generator devises data, the discriminator evaluates its veracity. Their duelling nature results in the generation of incredibly realistic synthetic data, transforming realms like art and design.

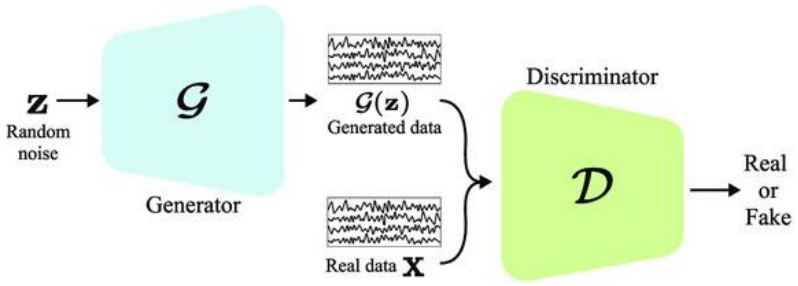

Figure 7: Illustration of Generative Adversarial Network (Goodfellow et al., 2014)

Shifting the lens to natural language processing, Attention Mechanisms and Transformers have emerged as pivotal innovations. Attention mechanisms, which underpin transformer architectures, weigh the importance of various parts of input data, making sequence processing more efficient. Models like BERT by (Devlin et al., 2018) and GPT by (Radford et al., 2018) have underscored the distinctive capabilities of transformers in capturing contextual information in language.

Furthermore, Radial Basis Function (RBF) Networks offer a distinctive approach in the neural network spectrum. These networks, explored by (Broomhead and Lowe, 1988), utilise radial basis functions as activation functions, making them particularly adept at function approximation and certain classification tasks. The essence of their operation rests on the distance between inputs and certain points in the input space, providing a unique paradigm in network processing.

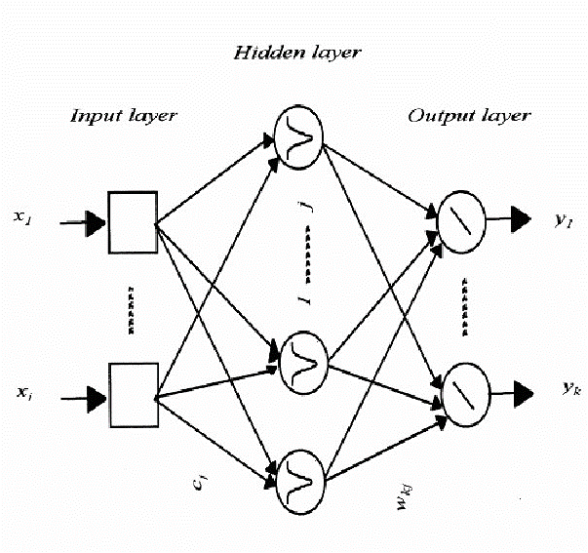

Figure 8: Radial Basis Function Neural Network architecture (Zang, Friswell and Imregun, 2003)

Finally, Self-Organizing Maps (SOMs), conceptualised by (Kohonen, 1982), cater to clustering and data visualisation tasks. Operating unsupervised, SOMs map high-dimensional data onto low-dimensional grids, preserving the topological characteristics of the original data. Their capacity to elucidate patterns in unlabelled data has made them instrumental in fields ranging from finance to biology.

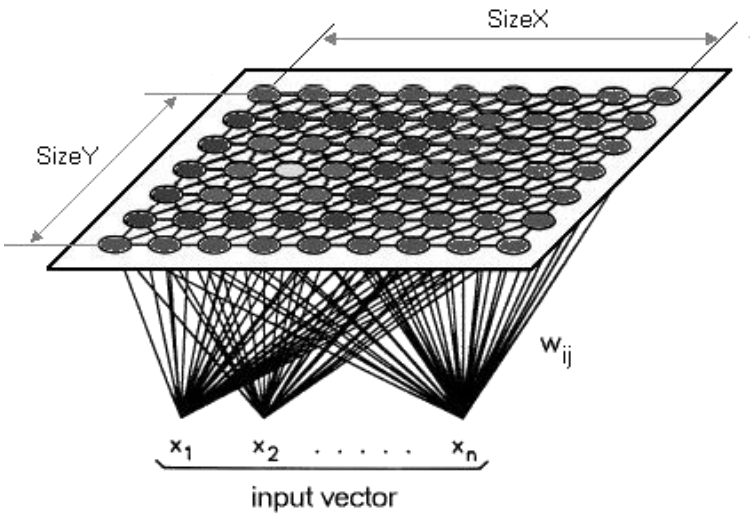

Figure 9: Illustration of Self-Organising Maps Kohonen Architecture (Eklavya, 2019)

Each of these architectures, while intrinsically rooted in the foundational doctrines of neural networks, unfolds its own narrative of evolution, challenges, and applications. The collective progress in these architectures chronicles the transformative evolution of neural networks and their ceaseless pursuit of mimicking and sometimes transcending human cognition.

2.5 Sparsity in Neural Networks

Sparsity pertains to minimising the quantity of non-zero parameters within a neural network. Sparse neural networks are frequently employed in order to enhance computational efficiency and reduce memory consumption. This is because sparse neural networks require smaller parameters for storage and processing (Dettmers and Zettlemoyer, 2019).

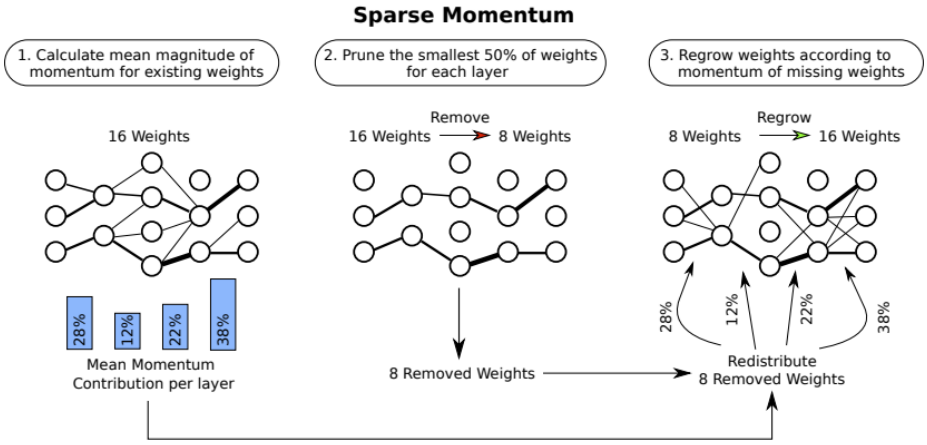

Figure 10: Illustration of Sparse Momentum (Dettmers and Zettlemoyer, 2019)

Previous studies have shown that employing basic heuristics that involve eliminating weights of small magnitude can result in significant compression rates while maintaining a minimal decrease in accuracy (Strom, 1997; Collins and Kohli, 2014). The study conducted by (Han et al., 2015) has demonstrated the effectiveness of magnitude pruning techniques in achieving compression rates and reducing computational complexity. Subsequent research by (Guo et al., 2016) has further improved the sparsification process, resulting in even higher compression rates and significantly reduced computational complexity.

Several techniques based on Bayesian statistics and information theory have been suggested by (Dai et al., 2018; Molchanov et al., 2017). These methods have successfully attained heightened compression rates while offering substantial theoretical justification and rationale. The system explores the associations between classical sparsification and regularisation techniques.

Several initial methods for reducing the complexity of neural networks involve utilising a second-order approximation of the loss surface to prevent the degradation of model performance (Hassibi and Stork, 1992). Recent studies have successfully attained similar compression levels using first-order loss approximations that are more computationally efficient. Additionally, further improvements have been made by establishing connections between these advancements and efficient empirical estimates of the Fisher information of the model parameters (Theis et al., 2018).

In various studies, reinforcement learning has been utilised in automated pruning weights and convolutional filters (He et al., 2018). Additionally, several approaches that are influenced by biological phenomena and evolutionary algorithms have been suggested (Guo et al., 2016; Mocanu et al., 2018).

One crucial aspect of a sparsity-inducing technique lies in its ability to impose structure on the topology of sparse weights, thereby influencing the overall characteristics of the technique. Although unstructured weight sparsity offers greater flexibility for the model, it presents challenges in efficient mapping to parallel processors and needs more support in deep learning software packages. Numerous methodologies have been developed to address the aforementioned concerns, emphasising eliminating entire neurons and convolutional filters and the imposition of block structure on the sparsely distributed weights (Gray et al., 2017). Although this approach has practical implications, a trade-off exists between the compression levels that can be achieved for a specific model quality and the degree of structure imposed on the weights of the model.

Several methods can be employed to achieve sparsity, including pruning, dropout, and regularisation techniques. Pruning involves removing connections based on specific criteria, often targeting the smallest weights. Dropout, however, randomly sets a certain proportion of weights to zero during the training process. Additionally, regularisation methods involve incorporating a penalty term into the loss function, which promotes small or zero weights.

2.6 Pruning in Neural Networks

The optimisation of deep neural network parameters through pruning has attracted significant attention due to its potential benefits in terms of time, memory, and energy efficiency, both during the training process and during testing. Pruning is a methodology employed in neural networks to enhance efficiency by eliminating superfluous connections, also known as weights, within the network. This phenomenon may result in a network that exhibits fewer connections while maintaining comparable performance.

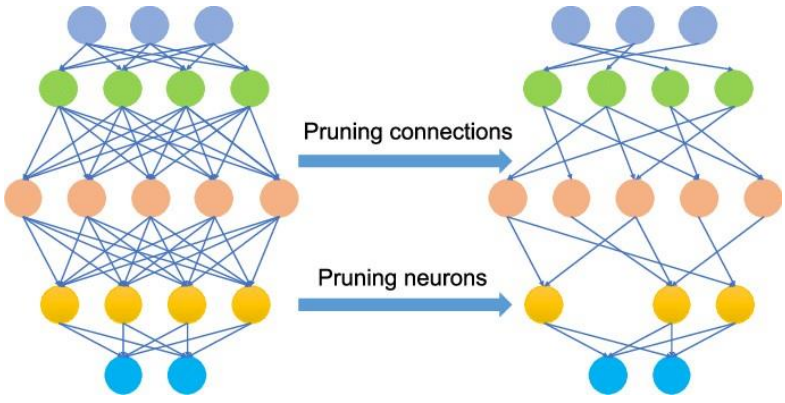

Figure 11: Illustration of pruning a neural network (Chen and Ran, 2019)

Traditionally, pruning algorithms have primarily concentrated on compressing models that have already been trained (LeCun et al.,1989; Han et al., 2015). However, recent studies (Frankle and Carbin, 2018) have shown that randomly initialised neural networks contain sparse subnetworks, also known as winning tickets. These winning tickets are determined through iterative training and pruning cycles, specifically through iterative magnitude pruning (Tanaka et al., 2020).

Various methods have been proposed for compressing neural networks, including developing innovative micro-architectures (Iandola et al., 2016), reducing network parameter dimensionality (Novikov et al., 2015), and training dynamic sparse networks.

2.6.1 Post-Training Pruning Paradigms

Traditional pruning algorithms commonly entail the assignment of scores to the parameters of neural networks after the training phase and removing the parameters with the lowest scores (Gale et al., 2019; Blalock et al., 2020; Tanaka et al., 2020). Commonly used scoring metrics in the field encompass weight magnitudes (Janowsky et al., 1989), the extension of these metrics to multi-layers (Park et al., 2020), the first-order Taylor coefficients, the second-order Taylor coefficients (Molchanov et al., 2019) of the training loss with respect to the parameters, and more advanced variations (Guo et al., 2016). Although the pruning algorithms can compress neural networks during test time, there is no decrease in the training cost.

Despite the advantages, a notable challenge associated with post-training pruning is the lack of clarity regarding the specific weights, neurons, or layers that should be eliminated. Another potential concern is the presence of ‘lottery tickets’, wherein certain subnetworks may exhibit higher effectiveness than others. However, identifying these subnetworks can pose a significant challenge. In addition, it should be noted that even networks that have been pruned may only partially capitalise on the potential advantages of sparsity. This is primarily due to the fact that state-of-the-art hardware is designed to optimise dense matrix operations, thereby restricting the extent to which speed improvements can be achieved through sparsity.

2.6.2 Precursor Training Pruning Paradigms

Recent research findings indicate that it is feasible to perform pruning on neural networks that have been randomly initialised before the training process. This pruning technique leads to a minimal or negligible reduction in the ultimate test accuracy (Lee et al., 2018). The Iterative Magnitude Pruning (IMP) algorithm, as described by (Frankle and Carbin, 2018),

involves a series of iterations that consist of training, pruning, and weight rewinding. This process aims to identify neural networks with high sparsity levels at initialisation while achieving comparable test accuracy to the original network. Despite the considerable computational capabilities of IMP, its usage requires repeating resource-intensive training and pruning procedures, which are further problematised by the requirement for accurate hyperparameter configurations.

To avoid these challenges, an alternative methodology leverages the gradients of the training loss during initialisation to perform network pruning in a unified manner (Lee et al., 2018). The single-shot pruning algorithms implemented during initialisation exhibit higher efficiency and comparable performance to IMP at moderate sparsity levels. However, they are susceptible to a phenomenon known as layer collapse, which involves the premature pruning of an entire layer, resulting in the network becoming untrainable (Lee et al., 2019).

2.7 The Distinction of the Lottery Ticket Hypothesis

The technique of neural network pruning is widely employed in order to decrease the dimensions of a trained model, thereby enabling efficient computational processes during inference while maintaining a minimal decrease in accuracy. Nonetheless, the approach mentioned above still necessitates training an over-parameterised network, as attempts to train a pruned network from the beginning have proven unsuccessful.

In a recent study conducted by (Frankle and Carbin., 2018), an interesting observation was made regarding the performance of pruned neural networks. When their weights were reset to their original values, it was found that these networks could still be trained to achieve favourable outcomes. Therefore, the authors proposed the ‘lottery ticket hypothesis’, which suggests that within a randomly initialised neural network, a subnetwork exists that, when trained independently, can achieve the same level of performance as the original network. This observation has acquired significant attention, prompting subsequent studies aimed at comprehending this state-of-the-art phenomenon. (Zhou et al., 2019; Ramanujan et al., 2019) have recently proposed algorithms that aim to identify subnetworks that exhibit high-performance levels. In recent studies conducted by (Zhou et al., 2019) and (Ramanujan et al., 2019), novel algorithms were introduced to identify subnetworks that exhibit satisfactory performance levels without training. According to (Ramanujan et al., 2019), an estimation was put forth suggesting that a neural network that is adequately over-parameterised and randomly initialised will contain a subnetwork that can achieve competitive accuracy comparable to that of the extensively trained network without undergoing any training process. The estimation above is a more robust iteration of the lottery ticket hypothesis.

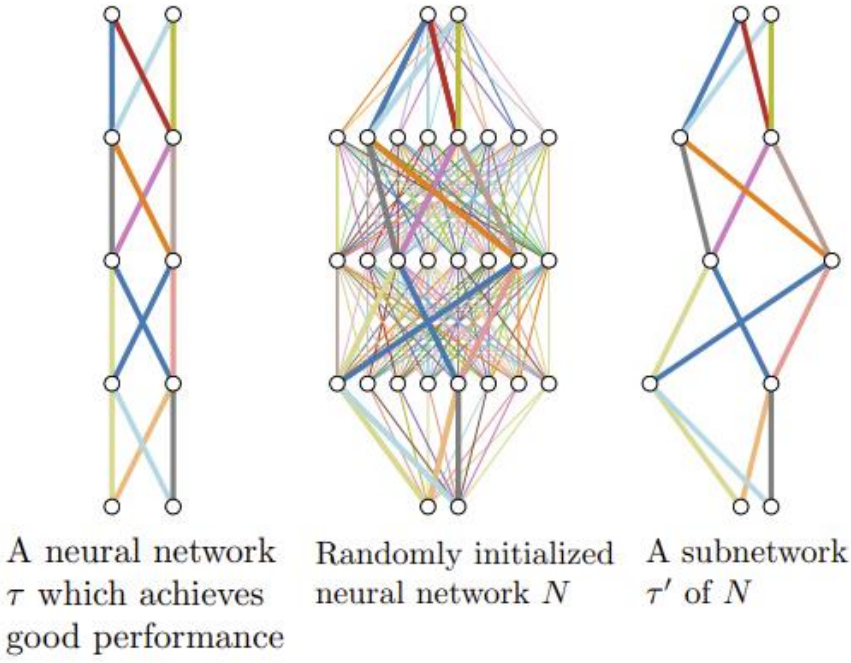

Figure 12: When a neural network has an abundance of random weights (centre), it possesses a subset (right) that can perform as effectively as a comprehensively trained neural network (left) with an analogous number of parameters (Ramanujan et al., 2020)

2.8 Continual Learning

2.8.1 Core Principles

When working in a scenario where data is presented in batches, neural networks have been shown to exhibit robust performance. Such frameworks concurrently grant the model access to the entire dataset, permitting methodical sampling and iterative revisiting to reinforce learning patterns. However, continual learning requires machine learning models to adapt continuously based on a consistent stream of novel data. Data instances are integrated sequentially in this scenario, aiming for an optimisation over all input data.

Drawing from foundational research by (Schwarz et al., 2018), (Hadsell et al., 2020), and (Delange et al., 2021), and building upon the framework provided by (Swaroop, 2022), an optimal lifelong learning system should exhibit several key characteristics:

- Limit the storage of historical data by imposing stricter restrictions to not store any past data or by considering factors such as memory and computational restrictions, as well as privacy concerns.

- Effectively manage the continuous presentation of new data examples without making assumptions about the underlying data structure. This means there are no predetermined tasks or datasets, and the boundaries between tasks are unclear.

- Neutralise catastrophic forgetting and interference, which refers to the aforementioned phenomenon where learning on novel data does not result in overwriting previously encountered data.

- Demonstrate the concept of forward transfer, which refers to the capacity to utilise prior knowledge to enhance performance on novel data, as well as the concept of backward transfer, which refers to the capacity to enhance efficacy on old data by utilising knowledge from new data.

- Ensure that the model's computational capacity either remains within set boundaries or exhibits effective scalability in response to an influx of data, guaranteeing the system's adaptability to prolonged data streams.

- The model should be able to sustain plasticity, which refers to its capacity to effectively acquire new knowledge and adapt as it encounters additional data.

The system should exhibit these characteristics while learning efficiently and maintaining a high accuracy. Achieving all of these desired properties in a machine-learning system is currently a problematic challenge. In an effort to achieve this objective, contemporary research generally focuses on fulfilling a specific set of these desired outcomes individually. This approach may be deemed justifiable, as various practical applications may necessitate distinct subsets of desired qualities.

Moreover, Figure 13 presents various paradigms for continual learning. Firstly, there are standard methods for nonsequential, multitask learning (A), where tasks are treated as independent and uniformly distributed. In these methods, multiple tasks are learned simultaneously, which helps prevent memory loss and maintain stability. Secondly, there are techniques rooted in gradients (B) that emphasise the importance of preserving parameters based on their relevance to previously learned tasks. Thirdly, strategies based on modularity (C) are employed to clearly delineate task-centric parameters. Often, these are placed alongside general parameters to promote knowledge transfer between tasks. Fourthly, there are approaches that use memory (D) as a foundation, wherein experiences are stored to combat the issue of forgetting. Lastly, meta-learning (E) stands out as a paradigm where the

focus is on refining ‘meta-goals’ for continual learning across a broad spectrum of task sequences, thereby optimising the process of continual learning.

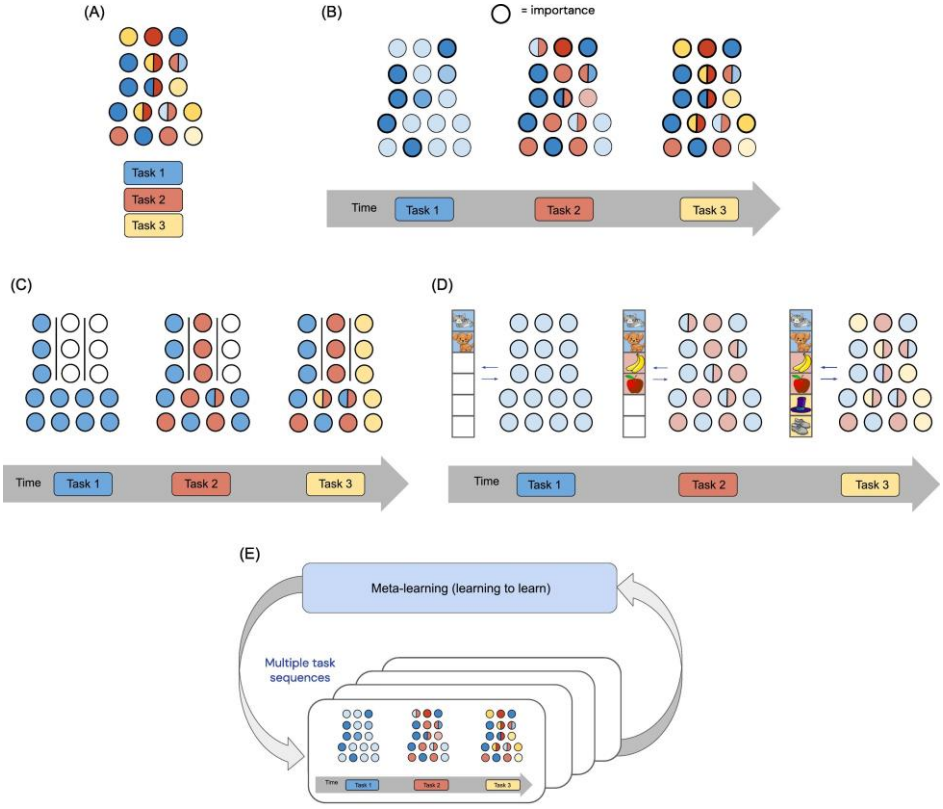

Figure 13: Paradigms for Continual Learning (Hadsell et al., 2020)

2.8.2 Relationship to Other Learning Scenarios

Within the academic literature, numerous learning scenarios have been developed to facilitate the efficient transfer of knowledge between datasets, especially under the constraints of data scarcity (Goodfellow, Bengio and Courville, 2016). In this section, we provide a comprehensive overview of their association with continual learning.

Multitask learning integrates information across multiple tasks simultaneously. It is essential to clarify that these tasks can manifest substantial heterogeneity, necessitating diverse loss functions, explicitly differentiating between classification and regression protocols (Caruana, 1997).

In contrast, transfer learning and domain adaptation capitalise on previously accrued knowledge from a designated source task, repurposing it to amplify efficacy on a subsequent

or target task. This paradigm allows for the utilisation of various source tasks with the primary objective of enhancing forward knowledge transfer while mitigating the potential setbacks of catastrophic forgetting (Pan and Yang, 2010).

In a similar context, few-shot learning demonstrates a rapid adaptability to a constrained set of instances originating from previously unencountered data classes. Concurrently, it operationalises strategies to suppress the adverse effects of catastrophic forgetting. A prevailing methodology in this domain integrates meta-learning algorithms on a substantially expansive dataset, preventing the exclusive reliance on online adaptation mechanisms (Wang et al., 2019).

Curriculum learning establishes the systematic progression of data input during the training phase of a learning model. The methodology involves initiating less convoluted instances, subsequently advancing the complexity gradient, thereby fostering enhanced model accuracy over the aggregate dataset (Bengio et al., 2009).

Furthermore, active learning exemplifies the algorithmic selection of specific data subsets from an extensive unlabelled dataset for targeted labelling. This methodology emerges in response to the functional constraints associated with obtaining supervised labels, necessitating an emphasis on the most crucial data points. Post-integration of this refined data, the algorithm undergoes iterative recalibration, setting the trajectory for subsequent data point inclusion (Settles, 2009).

In contrast, federated learning is established on a distributed data architecture wherein data remains partitioned across diverse client nodes. Owing to data integrity and security protocols which mandate data retention within its originating client ecosystem, there is an imperative for a communication-efficient algorithmic mechanism to link insights for the construction of an integrated model. The architectural plasticity permits recurrent and non-sequential access to each client's dataset. Federated learning, in essence, can be interpreted as a broad-based iteration of continual learning (Bui et al., 2018; McMahan et al., 2016).

With respect to continual learning, the foundational aim is the sequential assimilation of diverse computational tasks (Thrun, 1998) while navigating the complexities of catastrophic forgetting (French, 1999; Maes et al., 1996). Even with the increase of methodologies in the domain of continual learning, an evident gap persists in establishing a standardised architectural framework for training and evaluative metrics (Van de Ven and Tolias, 2019).

Distinctive parameters characterising diverse learning scenarios can be classified as 1) the presence of task identity during training, 2) the presence of task identity during inference, 3) the sharing of class labels during evaluation, and 4) the nature of the overall task matrix, whether it is discrete or continuous.

In addition, the existing strategies for preventing catastrophic forgetting can be roughly categorised into four groups: memory-based techniques, regularisation-based techniques, structure-based techniques, and hybrid techniques. The latter category integrates components from the former fundamental three categories, optimising their intrinsic strengths while addressing their potential individual limitations.

2.8.3 Contemporary Techniques

2.8.3.1 Memory-based Techniques

Techniques in this category prevent forgetting by explicitly or implicitly storing previous knowledge, with the latter serving as unprocessed samples. For example, (Chaudhry et al., 2018) aimed to empirically analyse the efficacy of a limited episodic memory within a continual learning framework, wherein each training instance is encountered only once. Remarkably, in continual learning, a primary baseline method that simultaneously trains on instances from the present task and instances stored in the episodic memory exhibits superior performance compared to specifically tailored continual learning approaches, both with and without episodic memory, across four distinct supervised learning benchmarks. In contrast, the latter approach involves using a generative model, such as a Generative Adversarial Network (GAN) (Shin et al., 2017) or an autoencoder (Kemker and Kanan, 2018) to synthesise the data for conducting pseudo-rehearsal. These techniques enable simultaneous training of multiple tasks on independent and identically distributed data, resulting in a substantial reduction in the occurrence of forgetting.

On the other hand, in a recent study conducted by (Chaudhry et al., 2019), various methods, including Gradient Episodic Memory (GEM) (Lopez-Paz and Ranzato, 2017), Meta-Experience Replay (MER) (Riemer et al., 2019), and Experience Replay Reservoir (ER-RES) (Chaudhry et al., 2019), were compared in the context of small episodic memories in continual learning. Like the approach employed by (Riemer et al., 2019), the ER-RES method utilised a reservoir sampling technique, which involved traversing the data in a single pass. In addition, the GEM framework employs an episodic memory that stores a subset of the observed samples. The detection of the forgetting case in this memory involves the measurement of the angle of the gradient vector and the evaluation of the proposed update. The efficiency of model updates is improved through the implementation of the Averaged

Gradient Episodic Memory (A-GEM) approach, which involves the modification of the loss function of GEM.

The Deep Generative Replay (DGR) approach, as described by (Shin et al., 2017), does not rely on the storage of previous samples. Instead, it employs a generative adversarial network (GAN) to generate synthetic data, thereby mitigating the problem of catastrophic forgetting. In this study, the implementation of a deep generative replay framework is introduced, which facilitates sequential learning across multiple tasks. This framework achieves this by generating and rehearsing synthetic data that closely resembles previously encountered training examples. The proposed model comprises a generator, and a solver functions as a repository of knowledge for a specific task. On the other hand, regularisation methods, such as Elastic Weight Consolidation (EWC) and the careful training of shared parameters, as seen in Learning without Forgetting (LwF), have demonstrated the potential to mitigate the issue of catastrophic forgetting by preserving prior knowledge within the network.

2.8.3.2 Regularisation-based Techniques

(Kirkpatrick et al., 2017) assumed that the learning capacity remains constant in the approaches above. Continual learning is then executed to regulate and minimise any alterations in the parameters, mainly if such changes result in a decline in performance on previous tasks. Hence, to select parameters, it is necessary to establish a concept of weight importance measurement to prioritise the utilisation of parameters. The author introduced a novel algorithm called Elastic Weight Consolidation (EWC) to tackle the substantial challenge of continual neural network learning. The use of Elastic Weight Consolidation enables the preservation of prior task knowledge while acquiring new knowledge, thereby preventing the occurrence of catastrophic forgetting of previously acquired abilities. This process is achieved by selectively reducing the plasticity of synaptic weights, which resembles neurobiological models of synaptic consolidation. The implementation of Elastic Weight Consolidation involves the utilisation of a soft, quadratic constraint. This constraint facilitates the adjustment of each weight towards its previous values, with the extent of adjustment being proportional to the weight's significance in achieving optimal performance on previously learned tasks.

Another study (Serra et al., 2018) presented Hard Attention to the Task (HAT), a robust attention mechanism that effectively preserves the information of primary tasks during the learning process of new tasks. This is achieved through task embedding, which enables the attention mechanism to concentrate on relevant task-specific features. In addition, the proposed hard attention mechanism is lightweight as it introduces only a minimal number of weights to the underlying network. Furthermore, it is trained concurrently with the primary

model using backpropagation and vanilla Stochastic Gradient Descent (SGD), resulting in negligible computational overhead. Next, the authors aimed to validate the efficacy of the proposed method for mitigating the issue of catastrophic forgetting in the domain of image classification. To achieve this, the researchers conducted comprehensive experiments using various datasets and employing state-of-the-art techniques.

In another study, (Ebrahimi et al., 2020) introduced a novel approach called Uncertainty-guided Continual Bayesian Neural Networks (UCB). The key feature of UCB is the adaptive learning rate, which is determined based on the uncertainty in the probability distribution of the weights within the neural networks. Uncertainty serves as a fundamental mechanism for discerning which information to retain and which to modify as part of an ongoing learning process, thereby mitigating the occurrence of catastrophic forgetting. Additionally, the author presented an alternative version of the model that incorporates uncertainty in the weight-pruning process. This variant ensures that task performance is preserved even after pruning, achieved by storing binary masks specific to each task. The UCB approach is thoroughly evaluated on various object classification datasets, encompassing both short and extended sequences of tasks.

2.8.3.3 Structure-based Techniques

These techniques leverage the concept of modularity and strive to confine the inference process within a specific network region, such as modules. On the other hand, the preservation of performance on previous tasks is achieved by storing the learned module. (Rusu et al., 2016) presents Progressive Neural Networks (PNNs) as a symbolic technique that showcases the static growth of architecture while simultaneously upholding lateral connections to previously frozen modules. This approach ensures zero forgetting but comes at the cost of a quadratic increase in the number of parameters.

(Li et al., 2019) introduced a novel framework referred to as the Learn to Grow. The present study presents a novel framework that effectively tackles the challenge of catastrophic forgetting in continual learning scenarios. This framework is characterised by its conceptual simplicity and broad applicability, making it a valuable contribution to the field. The methodology put forth in this study encompasses two discrete elements: one dedicated to the optimisation of the neural structure and another focused on the acquisition and/or fine-tuning of the parameters.

Additionally, the proposed method demonstrates the ability to evolve neural structures in a meaningful manner by separating explicit neural structure learning and parameter estimation. Furthermore, it exhibits strong capabilities in mitigating catastrophic forgetting in

experimental settings. Furthermore, the method demonstrates superior performance compared to all other baseline methods when evaluated on the permuted MNIST dataset and the partitioned CIFAR100 dataset. Even so, the effectiveness of the structure-based approach for continual learning is compromised by the substantial computational and memory requirements it involves.

Another study (Yoon et al., 2018) introduced a novel deep network architecture called the Dynamically Expandable Network (DEN) for lifelong learning. The DEN is designed to autonomously determine its network capacity during training on a series of tasks. This allows the DEN to acquire a concise and overlapping knowledge-sharing structure among the tasks. Despite the resulting computational cost, this strategy becomes critical in scenarios involving continual learning, wherein a substantial number of tasks need to be acquired, and the assumption of a fixed capacity cannot be made.

2.8.3.4 Hybrid Techniques

Hybrid approaches integrate various techniques from the aforementioned categories to harness their combined strengths and mitigate the limitations that each approach may have when implemented in isolation.

One hybrid approach is PathNet (Fernando et al., 2017), which combines structural and memory-based elements. In this method, a neural network is considered a shared parameter space where multiple paths can be formed. These paths, in turn, can be reused or fine-tuned for newer tasks, enabling a form of transfer learning. The key idea behind PathNet is to search for pathways in the neural network that can be utilised for new tasks without interfering with what has already been learned. The multi-agent competition is used to facilitate this search, ensuring the most efficient paths are selected for subsequent tasks.

Another notable work is Variational Continual Learning (VCL) (Nguyen et al., 2018). This method leverages a Bayesian approach, similar to the principles used in regularisation-based techniques, combined with a dynamic network architecture to adjust its capacity in a continual learning setting. VCL maintains a posterior distribution over the model's parameters and updates this distribution when new data is introduced. By preserving a distribution rather than point estimates, the model can better remember past tasks and be more prepared for new ones. The combination of Bayesian inference with dynamically adjusted network capacity makes VCL a robust hybrid approach.

Furthermore, (Hsu et al., 2018) presented a method which is a combination of structural and regularisation techniques. The authors acknowledged the importance of understanding when and how to apply continual learning, emphasising scenarios where tasks arrive in sequences, and the model might not have access to past data. By incorporating a structural framework with regularised learning, they proposed a method to revisit past tasks and adjust the learning procedure based on the model's confidence in its current knowledge.

The immediate advantage of hybrid techniques lies in their adaptability and resilience against catastrophic forgetting across a myriad of continual learning scenarios. By drawing on diverse strategies, these models often demonstrate higher robustness and flexibility, paving the way for more sophisticated continual learning paradigms in the future.

2.9 Supermasks in Superposition

The Supermask in Superpositions (SupSup) method is a robust approach that can effectively address the issue of catastrophic forgetting in continual learning settings. It utilises the Edge Popup algorithm (Ramanujan et al., 2020) to train supermasks, with the objective of minimising the cross-entropy loss L for each task k (Wortsman et al., 2020).

$$L = -\frac{1}{|\mathrm{X}_k|}\sum \log\, p(y \mid x, k) \tag{1}$$

Where X_k represents the training data for task k and

$$p(\, y \mid x, k\,) = f\big(h(x; \mathrm{W} \otimes \mathrm{M}_k)\big) \tag{2}$$

The symbol $\otimes$ is used to represent the element-wise product. The acquisition of masks is achieved through the identification of the top p% of entries within the score matrices. The p value is employed to evaluate the degree of sparsity demonstrated by the mask M_k (Kim et al., 2022). The subnetwork is identified by employing the Edge Popup algorithm (Ramanujan et al., 2020).

The methodology employed in our study is consistent with this structural approach, as it integrates task-specific supermasks. The Supermasks in Superposition (SupSup) methodology demonstrates a superior capacity to learn multiple tasks sequentially without

encountering catastrophic forgetting. The study employs a methodology that involves the utilisation of a randomly initialised, fixed-base network. The system proceeds to identify a subnetwork, known as a supermask, that exhibits commendable performance for each individual task. When task identity is provided during the testing phase, it becomes feasible to retrieve the relevant subnetwork while simultaneously minimising memory usage. In the absence of explicit task instructions, SupSup can employ gradient-based optimisation techniques to infer the task at hand. The aforementioned optimisation technique is employed to ascertain a linear combination of acquired supermasks that effectively reduces the output entropy.

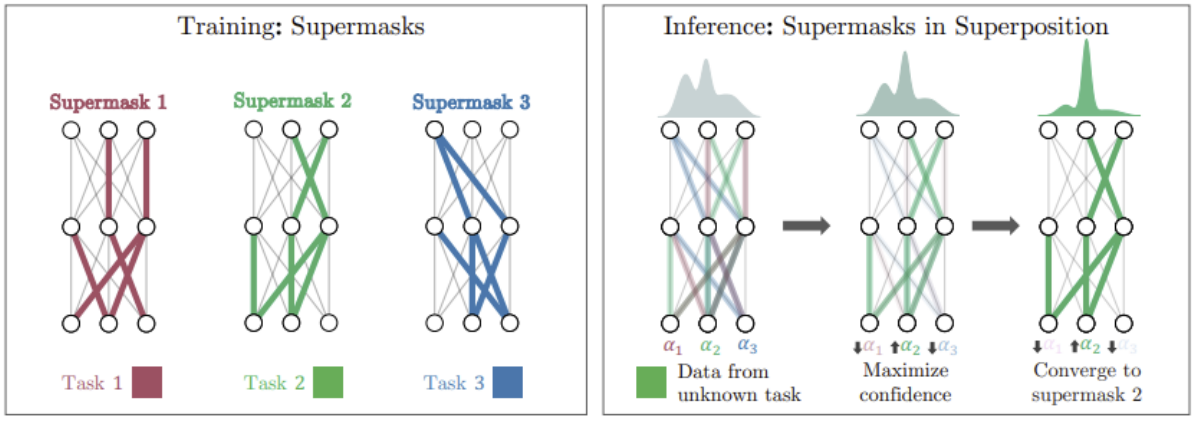

Figure 14: SupSup trains distinct supermasks for each task. (left). For inference, it combines supermasks with α_i weights and optimises confidence via gradients (right) (Wortsman et al., 2020).

The effectiveness of SupSup is considered to be marginally less than optimal as a result of the constraints imposed by fixed weights, which impede its capacity to accurately represent information. Furthermore, it should be noted that the model lacks the ability to accumulate or transfer knowledge during the process of acquiring new tasks. Moreover, it is crucial to consider the limitations that arise when applying SupSup, particularly in the context of inferring task identity, especially when tackling problems that are non-uniform and complex. The failure of task inference occurs when models do not possess appropriate calibration, resulting in an overestimation of their confidence in an incorrect task. Additional investigation is necessary to apply more calibrated models in the domain of automatic task inference. Moreover, there are difficulties related to calibration when utilising optimisation objectives like self-supervision and energy-based models.

In order to address this issue, (Yadav and Bansal, 2023) proposed the utilisation of Exclusive Supermask Subnetwork Training (EXSSNET). This innovative methodology prioritises the training of subnetwork weights that are exclusive and do not overlap with each other. In order to optimise performance and mitigate the risk of memory loss, a strategy is employed to prevent conflicting updates to the shared weights by subsequent tasks.

Furthermore, the author introduces a novel framework known as KNN-based Knowledge Transfer (KKT) that utilises existing knowledge to improve the efficiency and effectiveness of learning new tasks. Nevertheless, it has been observed that as the density of the mask increases, the degree of improvement in performance in relation to the SupSup method gradually decreases. The aforementioned phenomenon can be ascribed to the observation that denser subnetworks demonstrate higher levels of sparse overlap, consequently restricting the availability of free parameters for the purpose of updating new tasks.

Figure 15 demonstrates the schematic representation of EXSSNET where arbitrary weights, designated as W(0), are initiated. For the initial task, a supermask M1 is derived (visualised by the red subnetwork in column 2, row 1). Subsequently, weights associated with M1 are trained, and W(1) is produced (emphasised in bold red, column 1, row 2). In the ensuing task, mask M2 is extracted based on the previously established weights, W(1). If intersections between M2 weights and M1 are found (signified by the bold green dashed lines in column 3, row 1), only distinct weights (illustrated as unbroken green lines) for the task 2 subnetwork are updated (represented as bold continuous green lines in column 3, row 2). Weights that have been trained previously (shown in bold) are not altered in successive tasks. Concluding with task 3, mask M3 is delineated (represented in blue), and weights corresponding to it are accordingly adjusted.

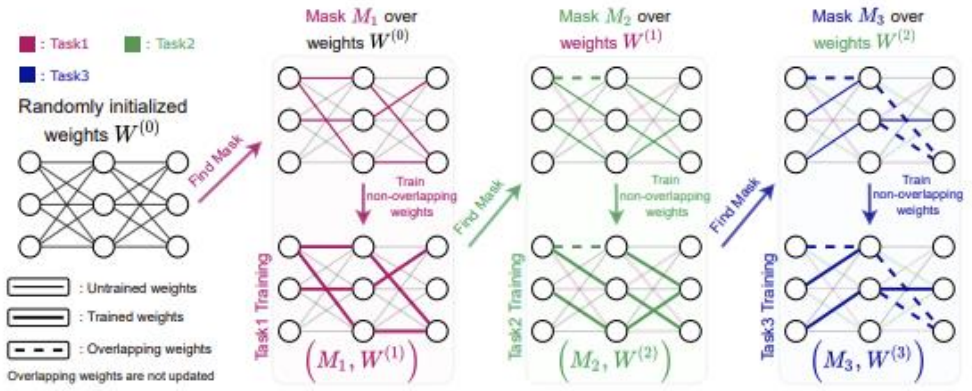

Figure 15: Illustration of EXSSNET methodology (Yadav and Bansal, 2023)

In another study, (Douillard et al., 2021) directed attention towards two prominent issues in the field of Continual Learning for Semantic Segmentation (CSS), specifically catastrophic forgetting and background shift. To tackle these concerns, the author has proposed a resolution referred to as Local POD, a distillation method that functions across various levels. The aforementioned technique successfully preserves the spatial statistics among pixels, including dependencies at both long and short distances. The observed phenomenon leads to the establishment of a balanced state between rigidity and plasticity in CSS, effectively reducing the likelihood of catastrophic forgetting. Subsequently, the

proposed system effectively tackled the problem of background shift by employing a loss function based on uncertainty-driven pseudo-labelling. The proposed methodology demonstrates a significant improvement in the enhancement of partially-labelled segmentation maps. This enhancement enables the neural network to effectively retain and utilise previously acquired knowledge. The author has successfully integrated rehearsal learning into CSS. This approach encompasses two distinct methodologies: partially labelled whole image rehearsal and object rehearsal, the latter of which demonstrates greater memory efficiency.

Extensive research has been dedicated to studying the process of acquiring knowledge from diverse input distributions, primarily focusing on the domain of classification. However, lifelong reinforcement learning (LRL) faces the additional challenge of adapting to changes in both the state and transition distributions, as well as the reward functions. The emergence of modulating masks in the realm of classification has demonstrated a significant level of flexibility in effectively tackling a wide range of task variations. The recent investigation conducted by (Ben-Iwhiwhu et al., 2023) focuses on the utilisation of modulating masks within the framework of deep LRL, specifically highlighting PPO and IMPALA agents. The effectiveness of this approach surpasses the performance of LRL baselines in both discrete and continuous reinforcement learning tasks. The research conducted a comprehensive examination of the utilisation of a linear combination of existing masks to effectively incorporate prior knowledge during the acquisition of new tasks. All iterations of the curriculum exhibit the capacity to acquire knowledge in a sequential manner without any notable decrease in information retention due to the introduction of distinct mask training.

2.10 Investigating Supermask Overlap for Enhanced Continual Learning

The focus of this study inquiry is to investigate the utilisation of supermasks within the framework of (Ramanujan et al., 2020; Wortsman et al., 2020) with the aim of improving the effectiveness of continual learning scenarios in neural networks. The main objective is to analyse the measurable degree of overlap between masks allocated to various tasks, thus revealing the possibility of accelerating the learning process. The attainment of this objective can be accomplished by conducting a thorough comparison of the binary masks on an element-wise basis (Zhou et al., 2019).

The computation of the sum of elements that possess a value of ‘1’ in both masks can be performed, and this resultant value can be normalised by dividing it by the total count of elements present in the mask. This would facilitate the measurement of the extent to which

the masks demonstrate overlap. If a specific level of similarity exists between tasks, it is reasonable to expect a higher degree of overlap in their corresponding supermasks. On the other hand, the potential exploitation of this convergence for more efficient learning can be approached in different ways.

One possible implication of overlapping supermarkets is the potential existence of shared representations across different tasks. If a cluster of weights (neurons) exhibits significance in multiple tasks, it implies that said neurons acquire representations that possess utility across various tasks. When a novel task exhibits similarity to prior tasks, utilising shared representations as an initial reference point can expedite the acquisition of new knowledge.

Furthermore, it is standard for the learning algorithm to extensively explore various potential configurations when initiating the training process of a neural network for a novel task. If there is a substantial degree of overlap among the masks in a task, it may be feasible to decrease the scope of the search; as per the findings of (Hossain et al., 2022), the learning algorithm has the capability to allocate greater importance to the weights that have been identified as ‘significant’ by the overlapping supermask for the specific task. Accordingly, this has the potential to decrease the amount of training time needed.

Chapter 3: Methodology

3.1 Overview

Continual learning, or lifelong learning, centres around a model's ability to accumulate knowledge over time, becoming adept in new tasks while maintaining proficiency in previously learned tasks. One of the critical challenges in continual learning is catastrophic forgetting, where models tend to overwrite or neglect what they have previously learned when exposed to new information or tasks, as exhaustively delineated in Chapter 2.

Recent advancements in the field, such as the work by (Wortsman et al., 2020), as delineated in Chapter 2, Section 2.9, present an intriguing methodology that, in a broad essence, incorporates the principles of continual learning and multitask learning (MTL). Their approach stands out as a representative paradigm that addresses the inherent challenges of continual learning.

At the core of their methodology is a randomly initialised, fixed base neural network where weights are shared across every task, which embodies the MTL principle of utilising shared representations. However, within the continuum of lifelong learning, the novelty lies in their introduction of task-specific binary masks, or the aforementioned supermasks, which overlay the shared weights. These supermasks dictate which weights are active for a given task. By learning these masks, the model can adapt to new tasks without disrupting the shared weight structure, thus counteracting catastrophic forgetting intrinsic to continual learning.

While MTL traditionally operates on the notion of training on multiple tasks simultaneously, continual learning is about handling tasks in a sequence, often with the constraint of not revisiting old data. The SupSup approach coordinates these principles (Wortsman et al., 2020). The shared weights capture the essence of MTL. At the same time, the task-specific masks ensure that the model can handle tasks sequentially without losing prior knowledge, catering to the continual learning paradigm. This incorporation of MTL and continual learning principles provides a balance between plasticity and efficiency. The shared weights offer a compact, efficient representation, while the supermasks ensure flexibility in adapting to new tasks without causing interference with the existing ones.

This chapter aims to showcase our methodology for this study based on the 'Supermasks in Superposition' framework to determine the extent of overlap between masks for different tasks and its consequent effect on collective task learning efficiency.

3.2 Models

3.2.1 Implementing Subnet Extraction and Weight Initialisation for Network Layers

3.2.1.1 Overview

Within the complex landscape of artificial neural networks (ANNs), meticulously controlling architecture and initialisation becomes a cornerstone for achieving optimal performance. This section elucidates the intricate methodologies we have utilised to extract specific subnets from a network layer and to methodically initialise the weights of those subnets. By doing so, we pave the way for a deeper understanding of the convoluted dynamics of our network, giving us more granular control over its behaviour.

3.2.1.2 Subnet Extraction

The capability to extract a specific subnet from a larger network layer is not just an advanced task but an essential one. This is especially true when there is a need to deeply analyse, modify, or fine-tune a particular segment of the neural network without perturbing the entirety of its structure. To cater to this requirement, we have utilised a specialised subclass within the autograd.Function called GetSubnet.

In the forward pass of this function, we applied straightforward binary thresholding to the scores. This is encapsulated in the static method forward, which takes the scores tensor and returns a binary tensor. An element in the returned tensor is set to 1 if the corresponding score is greater than or equal to 0; otherwise, 0. Mathematically, given a tensor of scores S, for every element s_i in S:

$$\text{result}_i = \begin{cases} 1 & \text{if } s_i \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

This can also be represented in a concise notation as:

$$\text{result} = \mathrm{H}(S) \tag{4}$$

Here, H symbolises the Heaviside step function, a mathematical function named after the renowned physicist Oliver Heaviside. The Heaviside step function can be visualised as a switch — it is zero for all values less than the threshold (in our case, zero) and one for all values greater than or equal to the threshold. Its application in neural networks, as we have employed here, allows for a clean binary distinction based on a given threshold, making it invaluable for tasks like subnet extraction. By utilising the Heaviside function, we have the ability to convert fluid, continuous scores into a definitive binary representation, which is pivotal when defining a subnet rooted in certain predefined criteria. This methodology of utilising step functions in neural networks has been explored and emphasised by pioneers in the field, including (Glorot and Bengio, 2010).

In the backward pass, the static method backward is overridden. In this implementation, the gradient g is returned as it is, meaning this function does not alter the gradient. This choice aligns with our intention of preserving the gradient information while extracting the subnet, ensuring that the extracted subnet maintains the gradient properties of the original network layer:

$$\frac{\partial \text{ loss}}{\partial S} = g \tag{5}$$

3.2.1.3 Weight Initialisation

Proper weight initialisation is a key aspect of training deep neural networks, affecting both convergence speed and the quality of the obtained local minima (He et al., 2015). We utilised two functions, mask_init and signed_constant, for this purpose.

The function mask_init creates and initialises each element of the ‘scores’ tensor that has the same shape as the ‘weight’ tensor of the given module. The scores are initialised with values drawn from a uniform distribution using the method nn.init.kaiming_uniform, with a scale factor $\sqrt{5}$:

$$\text{scores}_{ij} = \text{random}\ \left(-\sqrt{5}, \sqrt{5}\right) \tag{6}$$

Each element in the 'scores' tensor undergoes this initialisation. This method, named as the Kaiming or He Uniform initialisation, is tailored for deep networks predominantly using ReLU activation functions (He et al., 2015).

The function signed_constant calculates the correct fan-in for the given module, then calculates the gain for the ReLU activation function, and finally computes the standard deviation using the gain and the fan-in:

$$\sigma = \frac{g_{\text{ReLU}}}{\sqrt{f}} \tag{7}$$

Where g_{ReLU} is the gain for the ReLU activation function and f represents the fan-in for the given module.

Furthermore, the weights of the module w_{ij} are then updated by multiplying the $sign$ of the weight by the calculated standard deviation σ. The sign function is defined as:

$$\text{sign}\ (x) = \begin{cases} +1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases} \tag{8}$$

Using this definition, the weight updating process can be represented by:

$$w_{ij} = \text{sign}\ \left(w_{ij}\right) \times \sigma \tag{9}$$

This nuanced approach allows us to adjust the weights of the module in a principled manner, taking into account both the activation function and the architecture of the network (Glorot and Bengio, 2010).

3.2.1.4 Conclusion

In this section, we have presented a detailed analysis of the methods used to implement subnet extraction and weight initialisation for network layers. These methods are rooted in well-established principles and offer a robust and intuitive way to manipulate and control the behaviour of artificial neural networks.

Our approach to subnet extraction provides a flexible way to analyse and modify specific parts of the network, while our weight initialisation strategy is grounded in proven techniques that promote effective training of deep neural networks.

3.2.2 MultiTask Fully Connected Models with Masked Linear Layers (Classic and Novel)

3.2.2.1 Overview

The following section presents our methodology, which leverages the 'Supermasks in Superposition' framework to design multitask fully connected models. These models, augmented with masked linear layers, are central to our classic and novel approaches in handling multiple tasks in a continual learning setting.

3.2.2.2 Multitask Masked Linear Layers (Classic and Novel)

The MultitaskMaskLinear and MultitaskMaskLinearV2 classes serve as the cornerstones of our classic and novel approaches, acting as specialised linear layers capable of handling multiple tasks through masking techniques.

The classes extend PyTorch’s nn.Linear and introduce a new parameter num_tasks denoted as n to define the number of tasks to be handled. A list of parameters, scores, is initialised using a custom function mask_init denoted as f_{mask} , representing the task-specific masks. Mathematically, this initialisation can be represented as:

$$M_i = f_{\text{mask}}\,() \text{ for } i = 1 \text{ to } n \tag{10}$$

Where M_i represents task-specific masks for task i.

Furthermore, we disabled the gradients for the weight tensor, ensuring that the weight updates are solely governed by the subnet masks. To adjust the weights in alignment with our masking strategy, the function signed_constant f_{signed} is applied (Bengio, Léonard and Courville, 2013):

$$W = f_{\text{signed}}(W) \tag{11}$$

The method cache_masks is employed to cache the subnet masks without gradient tracking. This involves stacking subnets for all tasks and registering them as a buffer, a strategy commonly used for efficient memory management in neural networks (Pascanu, Mikolov, and Bengio, 2013). This can be formulated as:

$$M_T = [f_{\text{GetSubnet}}(M_1), f_{\text{GetSubnet}}(M_2), \dots, f_{\text{GetSubnet}}(M_n)] \tag{12}$$

Where M_T represents stacked masks of subnet for all tasks.

The forward propagation method 'forward' is employed with a focus on task specificity. This approach combines different subnets based on certain conditions, offering a dynamic adaptation to varying task requirements (Zhang et al., 2016). If the task index is less than 0, the subnet is computed using the following method:

$$M_T = \sum_{i=1}^{N} \alpha_i \times M_i \tag{13}$$

In this equation, M_T is the mask for a specific task, derived from the masks of previously learned tasks. Here, N stands for the total number of tasks that the system has learned. Each individual mask M_i from a previously learned task is multiplied by its corresponding coefficient α_i. These coefficients dictate how much influence each task-specific mask M_i on the resultant mask M_T. The values of α_i could be determined or adjusted based on the relevance or importance of each task in relation to the current task T. This formulation allows the system to adapt its behaviour by leveraging insights from tasks it has encountered previously.

Regarding task handling, in the case of a single task, the specialised class GetSubnet is applied to obtain the mask, reflecting our utilised modular approach to task handling. Moving on to weight transformation, the weight is multiplied by the subnet, followed by a linear transformation, reflecting the core idea of masked learning (Goodfellow, Bengio, and Courville, 2016). This is mathematically captured as:

$$W' = W \times M_T \tag{14}$$

$$y = F_{\text{linear}}\,(x, W', \text{bias}\,) \tag{15}$$

3.2.2.3 Multitask Fully Connected Network (Classic)

The MultitaskFC class, as presented in our methodology, establishes a comprehensive architecture using masked linear layers. This architecture leverages modern deep learning principles to offer a sophisticated approach to handling multiple (Ioffe and Szegedy, 2015; Nair and Hinton, 2010).

The architecture's foundation is the multitask masked linear layer with an input size of 784 and an output hidden size. For each task i, the output y_i from this layer is given by:

$$y_i = W_i \cdot x \odot M_i \tag{16}$$

Where W_i represents the weight matrix for the i^{th} task, x denotes the input vector, M_i is the mask matrix specific to the i^{th} task, and $\odot$ signifies element-wise multiplication.

Upon the completion of the linear transformation, the output is passed to a batch normalisation layer. This layer scales and shifts the data to ensure a zero mean and unit variance (Ioffe and Szegedy, 2015). The output y from the batch normalisation layer can be mathematically represented as:

$$y = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{17}$$

Where x is the layer's input, μ and σ^2 are the mean and variance of x, respectively, and ϵ is a small constant ensuring stability by avoiding division by zero.

Subsequent to batch normalisation, the output undergoes non-linear transformation using the ReLU activation function. The ReLU function introduces non-linearity, enabling the model to capture complex patterns in data (Agarap , 2018). The mathematical expression for ReLU is:

$$y = max(0, x) \tag{18}$$

This sequence—multitask masked linear layer followed by batch normalisation and then ReLU activation—is repeated twice in the model. The third and final sequence concludes with another masked linear layer with an output size of 100 but omits the batch normalisation and ReLU activation to prepare for the output layer.

Moreover, the get_bn_means method provides a systematic approach to tracking and extracting the internal statistics of the batch normalisation layer. For any given task, the method retrieves the mean for the associated batch normalisation layer. This is pivotal as it offers insights into the model's internal dynamics, serving as an invaluable asset for our subsequent analysis.

On the other hand, the get_masks method is dedicated to the retrieval and presentation of subnet masks from the MultiTaskMaskLinear layer. For a specified layer index, the method fetches the subnet masks. This is particularly crucial for our endeavours to visually and analytically comprehend the weight distributions across various tasks and to discern any task-specific features captured by the model, such as ours. To ensure accuracy and prevent potential errors, the method confirms that the layer in question indeed belongs to the MultitaskMaskLinear class.

Lastly, the forward method ensures the smooth transformation of input data through the model. It flattens the input tensor along dimension 1, allowing it to seamlessly pass through the established sequence of layers. The output from this method is the final result of the MultitaskFC class, ready for further processing and evaluation. This transformation can be mathematically represented as a series of nested functions:

$$y = f_{\text{Seq3}}\left(f_{\text{Seq2}}\left(f_{\text{Seq1}}(x)\right)\right) \tag{19}$$

Where each function f represents different components of the architecture. Specifically, f_{Seq1} and f_{Seq2} represent a combination of linear transformation, batch normalisation, and activation. Meanwhile, f_{Seq3} solely represents a linear transformation without the accompanying batch normalisation and activation.

3.2.2.4 Multitask Fully Connected Network (Novel)

The MultitaskFCV2 class encompasses an intricate architecture which is fundamentally identical to the MultitaskFC class with a core novel feature that elevates its capabilities.

In its core, the architecture builds upon the multitask masked linear layer with an input size of 784 and an output hidden size, denoted as MultitaskMaskLinearV2. For each task i, the output y_i from this layer is given by:

$$y_i = W_i \cdot x \odot M_i \tag{16}$$

Where W_i represents the weight matrix for the i^{th} task, x denotes the input vector, M_i is the mask matrix specific to the i^{th} task, and $\odot$ signifies element-wise multiplication.

Subsequent to this linear transformation, the output is ushered into a batch normalisation layer. This layer scales and shifts the data, ensuring that it adheres to a zero mean and unit variance. The transformed output y from the batch normalisation layer is represented mathematically as:

$$y = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{17}$$

Where x is the layer's input, μ and σ^2 are the mean and variance of x, respectively, and ϵ is a small constant ensuring stability by avoiding division by zero.

The output then undergoes a non-linear transformation through the ReLU activation function. This function introduces non-linearity into the model, empowering it to encapsulate intricate patterns within the data. The mathematical formulation for ReLU is:

$$y = max(0, x) \tag{18}$$

Similar to the classic approach, the architecture integrates this sequence—masked linear layer followed by batch normalisation and then ReLU activation—two times in succession. The third and final sequence concludes with another masked linear layer with an output size of 100, but, again, omits the batch normalisation and ReLU activation to prepare for the output layer.

A core novel feature is the set_alphas method. Alphas, denoted as α, are parameters that influence the behaviour of the MultitaskMaskLinearV2 layers. These parameters can be seen as scaling factors that modulate the emphasis on certain features or tasks. Given L as the set of layers in the model and α_l as the set of alpha values for layer l, this method facilitates the updating of alpha values as:

$$\alpha_l = \alpha_{new} \text{ for } l \in L \tag{20}$$

By dynamically adjusting these α values, we can guide the model's focus, thereby stressing the importance of task similarities or disparities. Such a mechanism can be instrumental in scenarios where tasks have inherent hierarchical or similarity-based relationships, allowing for nuanced control over learning dynamics.

The forward method remains the channel through which input data x undergoes transformation. The input data, upon entering the model, undergoes a sequence of transformations, layer by layer, until it emerges as the output y. This transformation can be mathematically represented as a series of nested functions:

$$y = f_{\text{Seq3}}\left(f_{\text{Seq2}}\left(f_{\text{Seq1}}(x)\right)\right) \tag{19}$$

In the architecture, each function f represents specific layers. Both f_{Seq1} and f_{Seq2} encompass a linear transformation followed by batch normalisation and activation. On the other hand, f_{Seq3} is limited to just the linear transformation, excluding batch normalisation and activation.

3.2.3 Conclusion

The architecture and methodology delineated herein represent our meticulous and innovative approach. By utilising specialised linear layers and a fully connected networks, we have architected a robust framework adept at managing multiple interrelated tasks. Moreover, both approaches are fundamentally identical as they rely on masked linear layers, disabling gradients for weight tensors, caching subnet masks, and employing similar forward propagation methods. In assessing the methodological differences between the classic and novel models, it is imperative to recognise the techniques unique to the classic approach and how they are subsequently augmented by the novel approach.

The classic approach places significant emphasis on the aforementioned methods, get_bn_means and get_masks. The get_masks method aids in retrieving and presenting subnet masks from the MultiTaskMaskLinear layer, playing a pivotal role in our attempts to determine the extent of mask overlap across various tasks. On the other hand, the get_bn_means method is instrumental in tracking and extracting internal statistics of the batch normalisation layer. For every task, this method retrieves the mean for the associated batch normalisation layer, thereby providing deeper insights into the model's internal dynamics and saving the extracted batch normalisation means for subsequent utilisation.

Building on these foundational insights, the novel approach leverages the extracted batch normalisation means from the classic approach to set the alphas for the masked linear layers of the novel model. This integration promises a potential targeted enhancement in learning efficiency, especially with the introduction of the modified alpha weights in the forward pass, showcasing a strategy deeply rooted in task similarity to potentially refine and accelerate the learning process.

These modifications in the classic and novel approaches are finely tuned to our research objectives of determining the extent of overlap between masks through task similarity and eventually leverage this overlap for faster learning. The core methods of the classic approach, which ascertain the extent of overlap between tasks, will be discussed in detail in the following section. Simultaneously, the convoluted techniques by which the novel approach utilises the batch normalisation means to determine the alphas and how they are subsequently managed will be comprehensively elaborated upon, ensuring a thorough understanding of both methodologies and their respective implications.

3.3 Supermask Overlap and Task Similarity Procedures

3.3.1 Overview

Understanding task similarities within a model's architecture is of paramount importance for harnessing shared information across tasks. This process facilitates more efficient learning by identifying and leveraging common features among various tasks. In this section, we delve into a rigorous analysis of the fundamental algorithmic and mathematical formulations used to determine the extent of overlap between masks for different tasks by calculating task similarities and determining the alphas (i.e., the coefficients used for task weighting), both integral to our classic and novel approaches.

3.3.2 Calculating Task Similarities using Jaccard Index (Classic)

The Jaccard similarity index, denoted by J, is a renowned measure historically introduced by the botanist Paul Jaccard in the early 20th century (Jaccard, 1901) to quantify the similarity between two sets. For any two sets, A and B, the Jaccard similarity index J is traditionally given by:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{21}$$

In the domain of data science, especially with binary data analysis, the Jaccard index stands as a seminal metric to gauge the overlap between binary vectors or sets (Real and Vargas, 1996). Considering our study context, wherein we engage with supermasks embodied as binary matrices, the aforementioned formula can adapted to:

$$J(S_1,S_2) = \frac{\text{Count of positions where both } S_1 \text{ and } S_2 \text{ are } 1}{\text{Count of positions where either } S_1 \text{ or } S_2 \text{ is } 1} \tag{22}$$

For a pair of supermasks of two different tasks, S_1 and S_2, the Jaccard index is evaluated as the quotient of the count of positions where both S_1 and S_2 possess a value of 1, to the count of positions where either S_1 or S_2 exhibits a value of 1.

The intersection between the two supermasks, S_1 and S_2, can be defined as:

$$intersection\ (S_1, S_2) = \sum_{i=1}^{n} S_{1i} \times S_{2i} \tag{23}$$

Where n is the total number of elements in the supermasks and S_{1i} and S_{2i} are individual elements of S_1 and S_2, respectively.

The union of the two supermasks can be represented as:

$$union\ (S_1, S_2) = \sum_{i=1}^{n} max(S_{1i}, S_{2i}) \tag{24}$$

Given the above definitions, the Jaccard index is then defined as:

$$J_{S_1,S_2} = \frac{\sum_{i=1}^{n} S_{1i} \times S_{2i}}{\sum_{i=1}^{n} max(S_{1i}, S_{2i})} \tag{25}$$

The resultant of this function is the Jaccard similarity index, presenting an invaluable quantification of the similarity between the supermasks of two distinct tasks. This metric is particularly appropriate for scenarios encompassing binary outcomes, such as our supermasks, establishing its significance in set similarity procedures.

3.3.3 Calculating Task Similarities using Cosine Similarity (Classic and Novel)

3.3.3.1 Classic Approach

In the classic approach, a function named calculate_task_similarityE1 was employed. The methodology began with the initialisation of a similarity matrix of zeros. If there are n tasks, the similarity matrix S can be represented as:

$$S = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}_{n \times n} \tag{26}$$

Following this, for each task, we looked to retrieve the mean from the first batch normalisation layer (specifically at index 1) of the task's batch normalisation means. Let's denote the mean retrieved from the batch normalisation layer of the i^{th} task as μ_i. The vector of means for all tasks can be represented as:

$$\mu = [\mu_1, \mu_2, \ldots, \mu_n] \tag{27}$$

Moving forward, the core phase was the calculation of the cosine similarity. For a given task i and another task j, the cosine similarity $\cos(\theta_{ij})$ is computed as:

$$\cos(\theta_{ij}) = \frac{\mu_i \cdot \mu_j}{\| \mu_i \| \| \mu_j \|} \tag{28}$$

This formula was employed to populate the similarity matrix S in line with the results of the calculation. It is crucial to mention that the cosine similarity function was instrumental in this step. Since it is well-regarded measure for determining similarity in high-dimensional spaces, as highlighted by (Mikolov et al. in 2013). As for the results, the function delivered a similarity matrix S. This matrix visually represented the pairwise similarities between the tasks, which were based on the means of the first batch normalisation layer.

3.3.3.2 Novel Approach

Building upon the classic approach, the novel approach utilised the function calculate_task_similarityE2. This methodology began with the preparation of a dictionary designed to store similarity matrices for diverse layers based upon the batch normalisation means extracted from the classic approach, as discussed in Section 3.2.3. Such an approach significantly augments the detail of similarity calculations.

A notable departure from the classic method is the layer-wise similarity calculation. For each layer l, the mean for task i can be represented as μ_{il}. Thus, the cosine similarity for this layer between task i and task j is:

$$\cos(\theta_{ijl}) = \frac{\mu_{il} \cdot \mu_{jl}}{\| \mu_{il} \| \| \mu_{jl} \|} \tag{29}$$

For every individual layer, a similarities matrix, initialised to zero, was established. The subsequent phase mirrored the classic approach. This was the calculation of cosine similarities, employing the same formula as mentioned above, after which the matrix was populated accordingly. The results of this approach were also distinct. The function offered a dictionary, and within this dictionary were similarity matrices tailored for different layers. This approach lends a more intricate insight into task affiliations, encompassing diverse phases of the model.

3.3.4 Determining Alphas in Layer-Wise Similarity (Novel)

The process of determining alphas is encapsulated within the function named determine_alphas, which is implemented in the novel approach. The methodology begins with the initialisation of a dictionary that is designed to hold alphas for various layers. For each layer, there exists a similarity matrix S where each element s_{ij} indicates the degree of similarity between task i and task j.

$$S = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1n} \\ s_{21} & s_{22} & \cdots & s_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n1} & s_{n2} & \cdots & s_{nn} \end{bmatrix} \tag{30}$$

As we delve deeper into the layer-wise determination of alphas, for each layer's similarity matrix, two primary steps are executed. Firstly, the task that exhibits the highest degree of similarity is identified. This is mathematically represented as:

$$j^* = \arg\max_j s_{ij} \tag{31}$$

This step is achieved by pinpointing the index that corresponds to the maximum similarity value for the current task at hand.

Following this identification, the alpha assignment is carried out. An alpha value of 1 is designated to the most similar task, while all other tasks are allocated an alpha value of 0. The mathematical representation of this assignment is:

$$A = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} \tag{32}$$

Where

$$\alpha_k = \begin{cases} 1 & \text{if } k = j^* \\ 0 & \text{otherwise} \end{cases} \tag{33}$$

The culmination of this function's operation is the return of a dictionary. This dictionary, denoted as D, is populated with the alpha values that pertain to different layers:

$$D = \{\text{task}_1 : A_1, \text{task}_2 : A_2, \dots, \text{task}_n : A_n\} \tag{34}$$

The significance of these alphas lies in their potential application: they can be employed to weight tasks. By doing so, emphasis is placed on the task with the highest similarity, which, according to (Yao and Doretto, 2010), can enhance the efficiency of the learning process.

3.3.5 Conclusion

Our supermask overlap and task similarity techniques aim to provide a deep insight into the nuances of understanding task similarities within our models' architecture. Determining the extent of task similarities lays the foundation for efficient learning, capitalising on shared information across various tasks.

Our analysis encompasses both classic and novel approaches. The classic approach, utilising the Jaccard similarity index, affirms the efficacy of this metric in ascertaining similarities between binary vectors such as our supermasks. Its historical prominence and adaptability to our domain demonstrate its continued relevance. Additionally, using cosine similarity offers a robust means of identifying task similarities in high-dimensional spaces.

The novel approach builds on the foundation laid by the classic method, introducing layer-wise similarity calculations. This approach offers an intricate understanding of task

affiliations throughout the model's architecture by providing a granular view of task similarities across different layers. This layer-specific insight is pivotal, as it aids in uncovering hidden patterns and affinities not discernible when analysing the model as an undivided construct.

Lastly, the novel approach of determining alphas illustrates a refined methodology for task weighting. The resultant alphas, emphasising tasks of higher similarity, can potentially elevate the learning process's efficiency.

3.4 Data Preliminary Analysis and Pre-processing

3.4.1 Original MNIST Dataset Overview

The digital revolution and the rise of big data have necessitated the use of efficient, automated systems to analyse and make sense of vast amounts of information. Central to these systems are machine learning algorithms, which learn to perform tasks by recognising patterns in the data they are given. Among the most frequently used databases in the machine learning domain is the Modified National Institute of Standards and Technology (MNIST) dataset, first introduced by (Lecun et al., 1998).

The MNIST dataset, comprising 70,000 greyscale images of handwritten digits, has been a foundational pillar in the realm of machine learning, particularly for computer vision tasks. Each image in this dataset, normalised to 28x28 pixels, is anti-aliased, meaning the greyscale pixels contain fractional intensity information, as illustrated in Figure 16. The high versatility of this dataset, coupled with its robust characteristics, makes it a powerful tool for diverse machine learning applications (LeCun et al., 1998).

Figure 16: Visual representation of the MNIST dataset: Handwritten digit samples (0-9)

This dataset was developed to facilitate the training and testing of machine learning algorithms, specifically for image recognition tasks. The arrangement of the dataset into a training set of 60,000 images, a test set of 10,000 images, 469 training batches, 79 validation batches and a batch size of 128 fosters a structured approach for the development and validation of novel algorithms (LeCun et al., 1998). The MNIST dataset, in virtue of its simplicity and clear structure, has been aptly referred to as the 'hello world' of machine learning, thereby serving as an essential stepping stone for novices in the field (Goodfellow, Bengio and Courville, 2016).

3.4.2 Prior Work

Over the past few decades, the MNIST dataset has played an instrumental role in some of the most significant advancements in the machine learning sphere. It has enabled researchers to investigate and address challenges related to catastrophic forgetting, which is a fundamental phenomenon deeply ingrained within artificial intelligence research, as comprehensively discussed in Chapter 2.

Trailblazing efforts with the MNIST dataset led to the advent of the Convolutional Neural Network (CNN) architecture known as LeNet-5, as illustrated in Figure 17 (LeCun et al., 1998). This groundbreaking architecture underscored the potent capabilities of CNNs in handling image recognition tasks and set the stage for future models. The MNIST dataset has since been extensively utilised by researchers worldwide to explore techniques to counteract catastrophic forgetting.

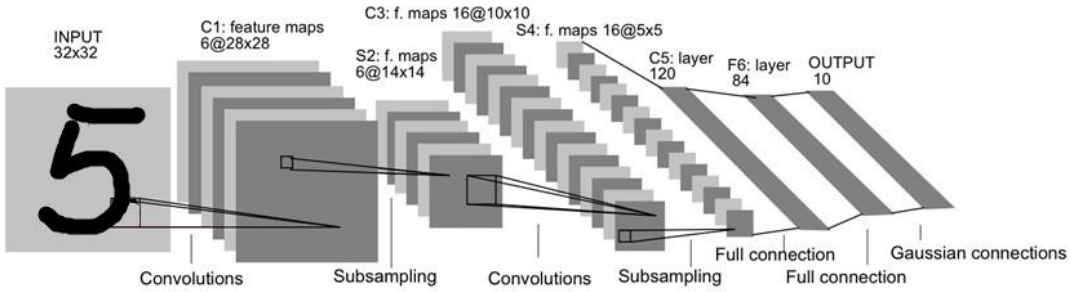

Figure 17: The architecture of LeNet-5 used for digit recognition for the MNIST dataset (Marques, Falcao and Alexandre, 2018)

One of the most noteworthy developments in this arena is the work by (Kirkpatrick et al., 2017). They proposed the previously mentioned algorithm Elastic Weight Consolidation (EWC), which directly tackled the issue of catastrophic forgetting. The EWC algorithm curtails catastrophic forgetting by imposing a quadratic penalty on the change of crucial weights of the neural network during the learning of new tasks, thereby safeguarding the performance of older tasks. The MNIST dataset was integral in validating this approach and demonstrating its value in setting a benchmark for assessing algorithmic progress.

Beyond its employment in standard classification tasks, the simplicity and structured format of the MNIST dataset make it a perfect testbed for a multitude of modifications and transformations. Variants, like permuted MNIST, rotated MNIST and partitioned MNIST, offer a series of related but distinct tasks, testing a model's ability to assimilate new knowledge while retaining previously learned information (Zenke, Poole, and Ganguli, 2017). These modifications have been leveraged to simulate a constantly changing environment, thereby testing the models against the adverse effects of catastrophic forgetting (Goodfellow et al., 2016).

Other pioneering research includes the work by (Finn, Abbeel and Levine, 2017), who proposed a Model-Agnostic Meta-Learning (MAML) approach. The MAML approach nurtures the development of algorithms that can swiftly adapt to new tasks. The MNIST dataset was key in validating these methods, further driving the pursuit of lifelong learning.

In the realm of generative models, the MNIST dataset is instrumental. Notably, Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), previously mentioned in Chapter 2, trained on MNIST, have demonstrated the ability to generate new data instances. These novel instances could have potential applications in augmenting continual learning by generating new training examples, thus diminishing the propensity towards catastrophic forgetting (Creswell et al., 2018).

Moreover, replay mechanisms and regularisation strategies, methods designed to mitigate catastrophic forgetting, frequently employ the MNIST dataset for testing. These techniques aim to strike a balance between the retention of old and new knowledge, which further extends the utility of the dataset in studying lifelong learning models (Robins, 1995; Zenke, Poole and Ganguli, 2017).

The MNIST dataset, despite its apparent simplicity, has been a cornerstone of machine learning research. It offers an excellent launchpad for understanding the mechanics of machine learning models and serves as a crucial tool for studying complex phenomena such as lifelong learning and catastrophic forgetting (LeCun et al., 1998). The broad usage and versatility of this dataset across various research areas underscore its significance and establish it as an integral component in the landscape of machine learning and artificial intelligence research.

3.4.3 Analysis of Class Distribution and File Sizes

The MNIST class forms the core of our investigation into the original MNIST dataset. We designed this class to initialise the dataset, defining a root path for data storage, and applying a series of transformations that are critical to the preprocessing phase. These transformations include the conversion of images into tensors and their subsequent normalisation. Such transformations are not arbitrary; they align with recognised practices within the deep learning community, encapsulating insights from leading researchers (Goodfellow et al., 2016).

Within the initialisation phase, we also made careful choices concerning data handling. We opted to use PyTorch's DataLoader, as it is optimised for large-scale data processing. The batch size was set at 128, based on empirical heuristics that have proven effective in numerous studies. Shuffling was specifically applied to the training data, a practice that has been shown to enhance the model's ability to generalise across unseen data (Bengio, 2012).

Furthermore, we provide a comprehensive overview of class distribution, both in terms of counts and percentages, enhanced with illustrative visualisations as depicted in Table 1 and Figure 18. Our dataset comprises ten unique classes. Within the training dataset, the number of images for each class varies between 5,421 and 6,742, while in the validation dataset, the count ranges from 892 to 1135.

Class Label	Training Counts	Validation Counts
0	5923	980
1	6742	1135
2	5958	1032
3	6131	1010
4	5842	982
5	5421	892
6	5918	958
7	6265	1028
8	5851	974
9	5949	1009

Table 1: Training and Validation Datasets Class Distribution – Counts

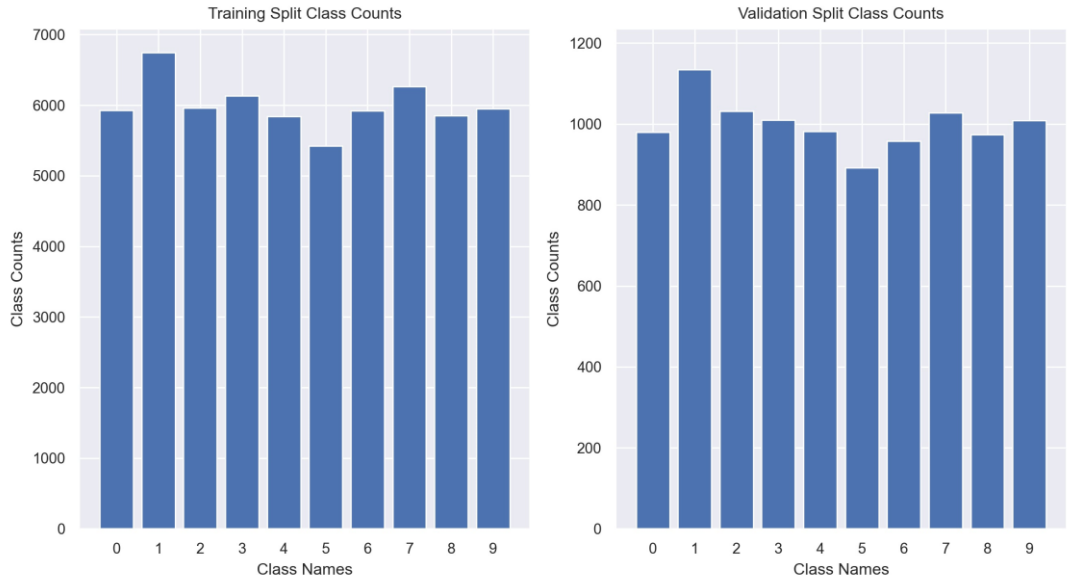

Figure 18: MNIST Dataset Training and Validation Class Distribution – Counts

In addition, to ensure a balanced dataset and mitigate potential biases, we thoroughly examined the class distribution and visualised the results as illustrated in Table 2 and Figure 19. Our analysis indicates a fairly consistent distribution across classes: in the training dataset, percentages span from 9.04% to 11.24%, and in the validation dataset, they range from 8.92% to 11.35%. Such a relatively balanced distribution is crucial as it prevents our model from favouring any specific class during training.

Class Label	Training Percentage	Validation Percentage
0	9.87	9.8
1	11.24	11.35
2	9.93	10.32
3	10.22	10.1
4	9.74	9.82
5	9.04	8.92
6	9.86	9.58
7	10.44	10.28
8	9.75	9.74
9	9.92	10.09

Table 2: Training and Validation Datasets Class Distribution – Percentages

Figure 19: MNIST Dataset Training and Validation Class Distribution – Percentages

After examining the class distribution, our focus shifted towards understanding the image file sizes within the MNIST dataset. Utilising a two-step methodology, we initially converted the tensor representations of images from the MNIST dataset into their standard image format, saving them in a consistent manner within a designated directory. This allowed for a more seamless analysis in our subsequent phase.

In the subsequent phase, a thorough evaluation of the saved images' file sizes was conducted, generating a comprehensive statistical overview of the dataset, as shown in Table 2. The dataset contained a total of 60,000 images, with the average file size hovering around 525.87 bytes. The variation in the sizes was evident, with a standard deviation of approximately 36.31 bytes, and the sizes ranged from as small as 393 bytes to as large as 658 bytes. Our quartile assessments further provided depth to this analysis: while 25% of the

images were up to 503 bytes in size, the median or the midpoint was established at 529 bytes, and a significant 75% of the images had sizes not exceeding 551 bytes.

Image size statistics	
Count	60000
Mean	525
Std	36
Min	393
25%	503
50%	529
75%	551
Max	658

Table 3: : MNIST Dataset image file statistics

3.4.4 MNIST Dataset Variants

3.4.4.1 Permuted MNIST

The concept of Permuted MNIST, as displayed in Figure 20, introduced us to an intriguing initial variation of the original MNIST dataset. This variant allowed us to see the effects of applying a consistent permutation to image pixel values, shedding light on its impact on the learning trajectory. These permutations, as established in past research (Kirkpatrick et al., 2017), are useful for assessing a model's sensitivity to the spatial alterations of pixels.

For this purpose, we designed the MNISTPerm class with an embedded sub-class dedicated to the permutation transformation. This architecture ensures a streamlined process where we can easily flatten the tensor, execute the permutation, and then reshape it. With this setup, we can effectively mimic how spatial modifications influence the model's learning.

Furthermore, we integrated a novel method for task adaptation. Within the realm of continual learning, it's pivotal to dynamically adjust and revert permutations based on task ID and seed. By incorporating these techniques, we not only capture the essence of real-world machine learning challenges (Parisi et al., 2019) but also pave the way for more adaptable learning models.

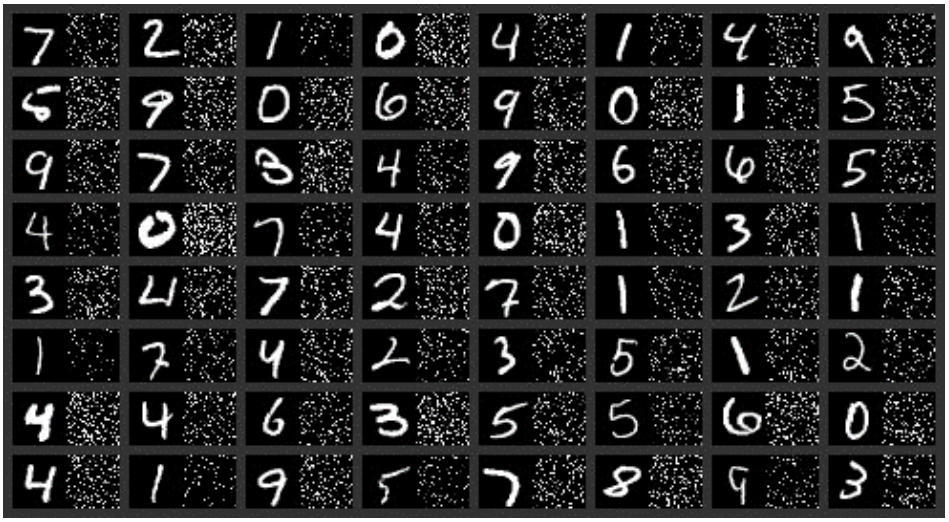

Figure 20: Display of MNIST handwritten digits (left) alongside their corresponding permuted versions (right), illustrating the complexity of recognising digits after pixel rearrangement

3.4.4.2 Rotated MNIST

The exploration of Rotated MNIST, as shown in Figure 21, provided us with an opportunity to examine how various orientations of the dataset could affect model performance. We designed methodologies to rotate images, providing insights into the model's robustness against changes in orientation.

Within the RotatingMNIST class, we constructed a callable object that enables the application of a specific rotation angle to an image. This rotation transformation served as a tool to explore orientation invariance, a property that has significant implications in object recognition (Anselmi et al., 2013).

Additionally, we utilised a method of introducing random rotations. Through this, by varying rotation angles randomly, we replicated the natural orientation fluctuations commonly found in visual data. This further enriched our insights into how models adapt to such unpredictable changes.

Figure 21: Three distinct variations of the Rotated MNIST dataset, showcasing the disparity and complexity introduced through digit rotation

3.4.4.3 Partitioned MNIST

Through our investigation and transformation of the Partitioned MNIST dataset, as illustrated in Figure 22, we gained deeper insights into the intricacies of domain adaptation. By segmenting the MNIST dataset into subsets, each featuring unique label pairs, we crafted distinct tasks. These tasks served as a platform to examine how models navigate different learning domains (Rebuffi, Bilen and Vedaldi, 2017).

In the PartitionMNIST class, we implemented a method to partition the dataset based on pairs of labels. This strategy yielded 10 distinct tasks, (0, 1), (2, 3), (4, 5), (6, 7), (8, 9), (0, 2), (1, 3), (4, 6), (5, 8), (7, 9), each signifying a unique combination of labels.

Furthermore, we utilised a dynamic task management approach. Our system is tailored to seamlessly switch between tasks, mirroring real-world instances where adaptive learning is paramount due to evolving conditions. This dynamic task handling is a critical component aligning with contemporary research trends.

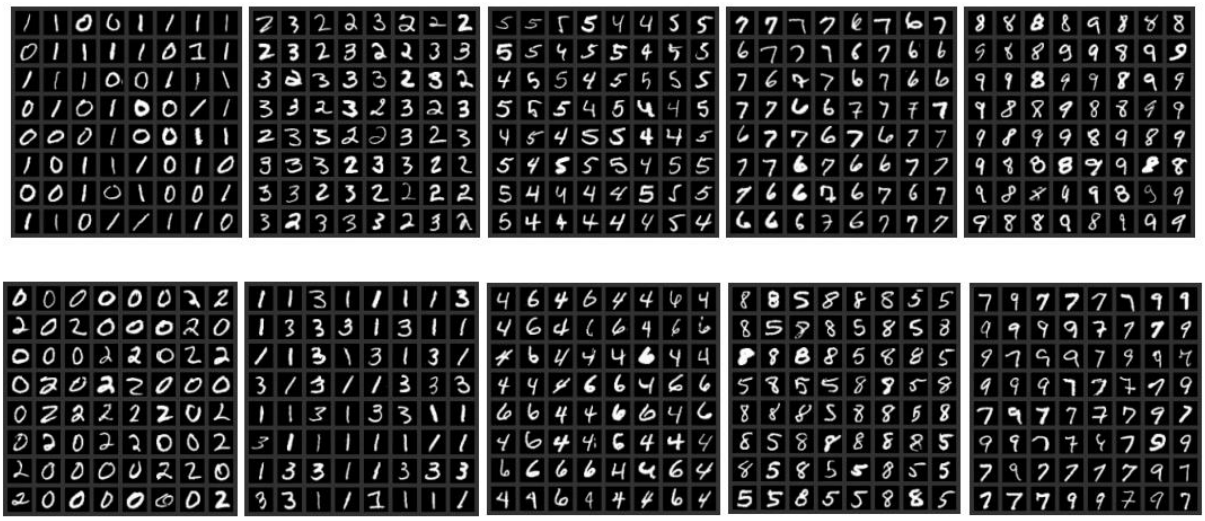

Figure 22: Illustration of the 10 Partitioned MNIST variations, each representing a unique label combination

3.4.5 Conclusion

The MNIST dataset, across its diverse variations, has been an indispensable instrument in our exploration of lifelong learning within neural networks. It equipped us with a framework to modulate learning conditions, assess model performance across varied contexts, and strategise against catastrophic forgetting. This section not only emphasised the pivotal role of the MNIST dataset in shaping and appraising machine learning models but also highlighted the importance of meticulous data preliminary analysis, preprocessing and transformation to fortify the credibility of our research.

3.5 Fundamental Training Functions

3.5.1 Overview

Training neural network models is a complex process. The core of this complexity lies in the iterative adjustment of model parameters, often denoted by θ. The primary objective is to minimise a specific loss function, L:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} L(\theta) \tag{35}$$

Such an iterative process is what grants deep models their learning capabilities. This process is central to the learning capabilities of deep models and has been the subject of extensive research (Bottou, 2010; Goodfellow et al., 2016).

In our study, we utilised two distinct training functions to align with the specific needs of both the classic and novel approaches. This section provides a detailed analysis of these training functions, illustrating the methodology and rationale behind each design.

3.5.2 Training Function for the Classic Approach

Our classic approach commences with the activation of the model's training mode, achieved through the model.train command. This activation ensures that specific layers operate as intended during training. For our loss function, we gravitated towards the Cross Entropy Loss, denoted as nn.CrossEntropyLoss. The mathematical representation of this loss function is:

$$L = -\sum_{i} y_i \log(p_i) \tag{36}$$

Here, y_i signifies the true labels, while p_i represents the predicted probabilities. This choice was influenced by its popularity and efficacy in classification tasks, as discussed by (Bishop, 2006). When engaging in our training loop, we navigate through the training data loader, processing both input data and corresponding labels in tandem. The model's forward method is then summoned to obtain logits, followed by the computation of the loss using our chosen criterion. The total loss calculated is then accumulated for further processing.

As we delve into gradient computation and optimisation, the standard backpropagation algorithm is put into action. This involves initialising the gradients to zero, calculating the gradients concerning the loss, and subsequently executing an optimisation step with our selected optimiser (Rumelhart, Hinton and Williams, 1986):

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t) \tag{37}$$

The η in the equation stands for the learning rate. It's worth noting that for our optimiser, we employed the RMSprop method with a learning rate of 1×10^{-4}, a choice driven by its

proven efficiency in numerous applications. A logging mechanism was seamlessly integrated into the training loop, set to display progress insights every 20 steps. These insights encompass the current epoch, step, loss, and accuracy metrics.

Towards the end of our classic approach, we invoked MultitaskFC's get_bn_means method to procure the batch normalisation means of the current task. The culmination of the training process involves the calculation of the average loss for the entire epoch, computed as:

$$\text{Average Loss} = \frac{\text{Total Loss}}{\text{Number of Batches}} \tag{38}$$

The function then returns the epoch's average loss along with the batch normalisation means for the designated task.

3.5.3 Training Function for the Novel Approach

Our novel approach, encapsulated within the trainV2 function, showcases striking resemblances with the classic methodology. The model is primed for training in an identical fashion, and the Cross Entropy Loss remains our loss function of choice. Furthermore, subsequent steps of loss computation, gradient computation, and optimisation mirror the classic approach. The logging mechanism remains consistent, providing periodic updates on training progress. As the loop concludes, the average loss for the epoch is computed similarly to the classic approach:

$$\text{Average Loss} = \frac{\text{Total Loss}}{\text{Number of Batches}} \tag{38}$$

Similarly, the function then seamlessly returns the average loss for the epoch.

3.5.4 Conclusion

The carefully employed training functions for both the classic and novel approaches epitomise the intricate requirements of our models training procedures. These functions not

only reflect the complex integration of various components during our neural network training but also significantly support our core research objectives.

The two training functions share a foundation of common methodologies. Both harness the power of the Cross Entropy Loss and employ a standardised training loop with forward and backward passes, enriched with logging capabilities. However, a minor divergence is evident. The classic approach training function incorporates an additional layer of complexity by handling and returning the extracted batch normalisation means, a feature explicitly absent in the novel approach's training function.

3.6 Performance Evaluation

In contemporary machine learning research, the utilisation of comprehensive evaluation metrics to assess the performance of multi-class classification models is imperative. The objective is to establish a rigorous comparison of models' capabilities, thereby informing both research directions and practical applications.

The evaluation procedure, utilised to assess a model's performance on a validation dataset, leverages the capabilities of PyTorch. The inclusion of the torch.no_grad decorator, as highlighted by (Paszke et al., 2019), underscores the exclusion of gradient computations during this evaluative phase. Upon initialisation, the model adopts its evaluation mode, ensuring that specific layers function in an optimised manner (Ioffe and Szegedy, 2015).

During the evaluation, the model's predictions are inferred from the logits, ascertained through the forward pass. The predicted class, P, can be discerned through the equation:

$$P = \text{argmax}\,(L) \tag{39}$$

Where L denotes the logits. Subsequent to this inference, accuracy calculation provide a quantified perspective on the model's performance:

$$Accuracy = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \tag{40}$$

The prediction procedure, structured to extrapolate the model's predictions on our designated dataset, also incorporates the torch.no_grad decorator. Post initialisation, wherein the model is set to its evaluation mode, lists are instantiated to archive the model's predictions and corresponding ground truths.

In striving for methodological rigour, we have opted for a suite of performance metrics extensively acknowledged in academic literature: precision, recall, and F1-score. These metrics serve as integral instruments in gauging a model's aptitude in classification tasks. However, in the realm of multi-class classification, the amalgamation of these metrics necessitates discernment, especially with accuracy's potential pitfalls in imbalanced datasets.

Two predominant strategies in multi-class classification are commonly utilised, micro-averaging and macro-averaging. While the micro-averaging approach may inadvertently prioritise frequent classes, the macro-averaging strategy computes metrics for each class in isolation, subsequently averaging the results.

Given the evident relative class disparities in our dataset, as shown in Section 3.4, our inclination towards the macro-averaging strategy emerges from the commitment to providing an unbiased performance assessment. Thus, we evaluate the performance of our models based on the macro-averaged precision, recall, and F1-score.

The first metric, precision, is defined as:

$$\text{Precision}_{\text{macro}} = \frac{\sum_{i=1}^{C} \frac{TP_i}{TP_i + FP_i}}{C} \tag{41}$$

Where C represents the total number of classes, TP_i denotes True Positives and FP_i stands for False Positives.

The second metric, recall, is defined as:

$$\text{Recall}_{\text{macro}} = \frac{\sum_{i=1}^{C} \frac{TP_i}{TP_i + FN_i}}{C} \tag{42}$$

Here, FN_i denotes False Negatives. Finally, the F1-score, a harmonic mean of precision and recall, is given by:

$$F1 - \text{Score}_{\text{macro}} = \frac{\sum_{i=1}^{C} \frac{2 \times \text{Precision}_{\text{macro}_i} \times \text{Recall}_{\text{macro}_i}}{\text{Precision}_{\text{macro}_i} + \text{Recall}_{\text{macro}_i}}}{C} \quad (43)$$

The performance evaluation procedures described are critical in the model development process. By adhering to established methodologies, these procedures provide a reliable means to assess our models' performance.

3.7 Conclusion

The methodologies presented in this chapter offer a robust and comprehensive framework to facilitate the experiments of our study. For model development, the masked linear layer allowed selective parameter sharing between tasks. Both the classic and novel model architectures built upon this with fully connected networks, batch normalisation, and ReLU activations. The novel architecture introduced adjustable alpha parameters to leverage task similarities.

Furthermore, the procedures for calculating task similarity using Jaccard and cosine similarity indices enabled quantifying the extent of overlap between masks. This knowledge could then guide the tuning of alpha parameters in the novel architecture to stress commonalities and potentially accelerate the learning process.

The preliminary analysis and preprocessing of the MNIST dataset provided the foundation, giving insights into class distributions, image properties and generating dataset and task variations through techniques like permutation, rotation and partition.

Lastly, the tailored training functions, alongside performance evaluation metrics like macro-averaged precision, recall and F1 score, completed the toolbox for our further rigorous experimentation and analysis.

Chapter 4: Experiments

4.1 Experiment 1

4.1.1 Overview

In this experiment, the aim was grounded on the exploration of learned supermasks across different tasks on distinct variations of the MNIST dataset and the assessment of their overlap through the Jaccard similarity index delineated in Section 3.3.2. The experiment encompassed the three specific variants of the MNIST dataset: Permuted MNIST, Rotating MNIST, and Partitioned MNIST.

4.1.2 Experimental Procedures

The experiment was structured in a methodical manner, ensuring consistency across all three MNIST variants. Each variant of the MNIST dataset was initialised, setting the number of tasks to 10. Subsequently, our classic multitask fully connected model, MultitaskFC, was initialised. This model's hidden layer size was set to 300 and it was tasked to handle 10 different tasks.

For each task within the chosen MNIST variant, the model was set to the current task, employing the set_model_task function. The dataset then updated its focus to the current task with the update_task function. An optimiser, specifically RMSprop, was employed with a learning rate of 1×10^{-4}. The model underwent training for a single epoch, followed by an evaluation on the dedicated validation set. Post this training, the supermasks pertinent to the current task were extracted and systematically stored in dedicated lists for each dataset variant.

A significant portion of the experimental procedure was dedicated to the overlap analysis between supermasks. Utilising the Jaccard similarity index, the overlap between every pair of supermasks was computed. The calculated overlap was then visualised in a three-dimensional scatter plot, where the axes represented the tasks being compared and the overlap percentage between their respective supermasks.

The procedure also emphasised the visualisation of individual supermasks. Each supermask, across all tasks, was depicted using the plot_supermask function, presenting them in a structured 5x2 grid for comprehensive visualisation.

Lastly, a rigorous statistical analysis was executed. A t-test was performed on the supermasks of all task pairs to ascertain statistical significance. The ensuing results, including the t-statistic and p-value, were systematically catalogued in a dataframe. This dataframe was then rearranged, ranking task pairs based on their similarity, determined by the p-value.

4.1.3 Experimental Results

4.1.3.1 Permuted MNIST

For the Permuted MNIST variant, the Jaccard index overlap for supermasks across tasks obtained and subsequently visualised via a 3D scatter plot are illustrated in Table 4 and Figure 23.

Furthermore, the individual neural activity of the supermasks for all tasks within this variant is graphically represented, as shown in Figure 24. In addition, a tabulated representation of the t-test results for all task pairs is showcased, which is sorted by highlighting task pairs from the most similar to the least similar based on p-values in Table 5.

Task ID	**Task ID**	**Overlap**
Task: 0	Task: 1	Overlap: 33.12%
Task: 0	Task: 2	Overlap: 32.75%
Task: 0	Task: 3	Overlap: 32.76%
Task: 0	Task: 4	Overlap: 33.15%
Task: 0	Task: 5	Overlap: 33.07%
Task: 0	Task: 6	Overlap: 33.13%
Task: 0	Task: 7	Overlap: 32.98%
Task: 0	Task: 8	Overlap: 33.06%
Task: 0	Task: 9	Overlap: 32.96%
Task: 1	Task: 0	Overlap: 33.12%
Task: 1	Task: 2	Overlap: 33.23%
Task: 1	Task: 3	Overlap: 32.82%
Task: 1	Task: 4	Overlap: 32.98%
Task: 1	Task: 5	Overlap: 32.87%
Task: 1	Task: 6	Overlap: 33.19%
Task: 1	Task: 7	Overlap: 32.84%
Task: 1	Task: 8	Overlap: 32.93%
Task: 1	Task: 9	Overlap: 32.91%
Task: 2	Task: 0	Overlap: 32.75%
Task: 2	Task: 1	Overlap: 33.22%

Task: 2	Task: 3	Overlap: 32.98%
Task: 2	Task: 4	Overlap: 33.02%
Task: 2	Task: 5	Overlap: 32.66%
Task: 2	Task: 6	Overlap: 32.79%
Task: 2	Task: 7	Overlap: 32.94%
Task: 2	Task: 8	Overlap: 33.05%
Task: 2	Task: 9	Overlap: 32.92%
Task: 3	Task: 0	Overlap: 32.76%
Task: 3	Task: 1	Overlap: 32.82%
Task: 3	Task: 2	Overlap: 32.98%
Task: 3	Task: 4	Overlap: 33.09%
Task: 3	Task: 5	Overlap: 32.66%
Task: 3	Task: 6	Overlap: 33.35%
Task: 3	Task: 7	Overlap: 33.06%
Task: 3	Task: 8	Overlap: 33.02%
Task: 3	Task: 9	Overlap: 32.96%
Task: 4	Task: 0	Overlap: 33.15%
Task: 4	Task: 1	Overlap: 32.98%
Task: 4	Task: 2	Overlap: 33.02%
Task: 4	Task: 3	Overlap: 33.09%
Task: 4	Task: 5	Overlap: 32.87%
Task: 4	Task: 6	Overlap: 33.04%
Task: 4	Task: 7	Overlap: 32.93%
Task: 4	Task: 8	Overlap: 33.26%
Task: 4	Task: 9	Overlap: 33.09%
Task: 5	Task: 0	Overlap: 33.07%
Task: 5	Task: 1	Overlap: 32.87%
Task: 5	Task: 2	Overlap: 32.66%
Task: 5	Task: 3	Overlap: 32.66%
Task: 5	Task: 4	Overlap: 32.87%
Task: 5	Task: 6	Overlap: 33.02%
Task: 5	Task: 7	Overlap: 32.89%
Task: 5	Task: 8	Overlap: 32.82%
Task: 5	Task: 9	Overlap: 32.86%
Task: 6	Task: 0	Overlap: 33.13%
Task: 6	Task: 1	Overlap: 33.19%
Task: 6	Task: 2	Overlap: 32.79%
Task: 6	Task: 3	Overlap: 33.35%
Task: 6	Task: 4	Overlap: 33.04%
Task: 6	Task: 5	Overlap: 33.03%
Task: 6	Task: 7	Overlap: 33.04%
Task: 6	Task: 8	Overlap: 32.74%
Task: 6	Task: 9	Overlap: 33.14%
Task: 7	Task: 0	Overlap: 32.98%
Task: 7	Task: 1	Overlap: 32.84%
Task: 7	Task: 2	Overlap: 32.94%
Task: 7	Task: 3	Overlap: 33.06%
Task: 7	Task: 4	Overlap: 32.93%
Task: 7	Task: 5	Overlap: 32.89%
Task: 7	Task: 6	Overlap: 33.04%
Task: 7	Task: 8	Overlap: 32.87%
Task: 7	Task: 9	Overlap: 33.21%
Task: 8	Task: 0	Overlap: 33.06%
Task: 8	Task: 1	Overlap: 32.94%
Task: 8	Task: 2	Overlap: 33.05%
Task: 8	Task: 3	Overlap: 33.02%
Task: 8	Task: 4	Overlap: 33.26%

Task: 8	Task: 5	Overlap: 32.84%
Task: 8	Task: 6	Overlap: 32.74%
Task: 8	Task: 7	Overlap: 32.87%
Task: 8	Task: 9	Overlap: 33.09%
Task: 9	Task: 0	Overlap: 32.96%
Task: 9	Task: 1	Overlap: 32.92%
Task: 9	Task: 2	Overlap: 32.95%
Task: 9	Task: 3	Overlap: 32.96%
Task: 9	Task: 4	Overlap: 33.09%
Task: 9	Task: 5	Overlap: 32.86%
Task: 9	Task: 6	Overlap: 33.14%
Task: 9	Task: 7	Overlap: 33.21%
Task: 9	Task: 8	Overlap: 33.09%

Table 4: Illustration of the overlap percentage between the supermasks of 10 task pairs

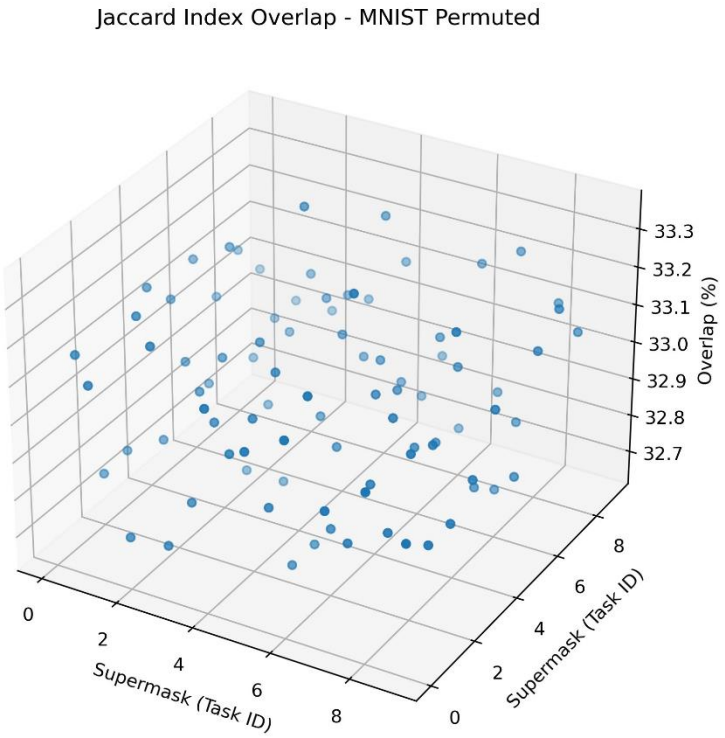

Figure 23: Visual representation of the overlap percentage between the supermasks for 10 task pairs

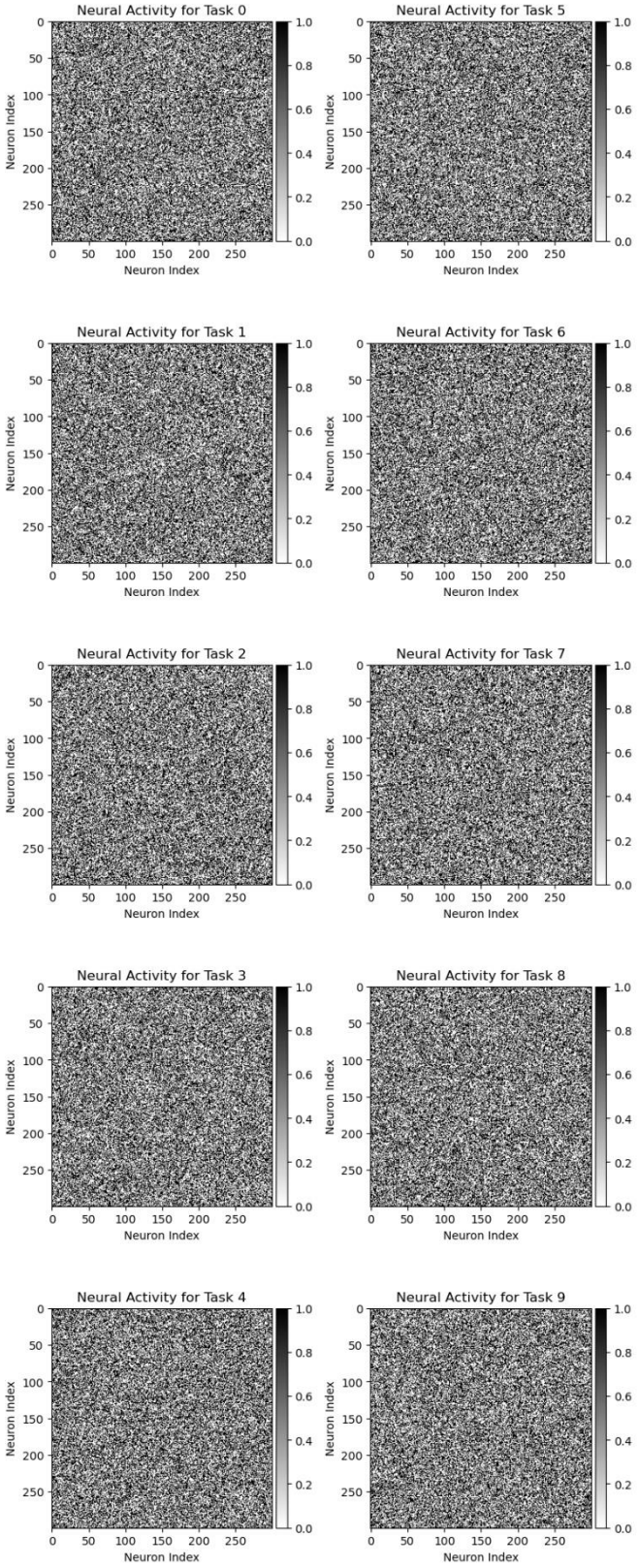

Figure 24: Visual illustration of the neural activity patterns across different tasks. Each subplot represents the neural activity for a specific task, with darker shades indicating higher activity

Task Pair	t-statistic	p-value
(3, 8)	-0.070711	0.943628
(2, 7)	0.080141	0.936125
(1, 2)	0.108427	0.913657
(0, 9)	-0.122565	0.902452
(7, 9)	0.14143	0.887534
(1, 7)	0.188568	0.850432
(3, 5)	0.207444	0.835663
(2, 9)	0.221571	0.824648
(0, 7)	-0.263995	0.791784
(5, 8)	-0.278156	0.780893
(1, 9)	0.329998	0.741402
(0, 2)	-0.344136	0.730744
(0, 8)	0.443153	0.657656
(0, 1)	-0.452563	0.650864
(4, 6)	0.471409	0.637349
(0, 3)	0.513864	0.607348
(8, 9)	-0.565718	0.571586
(3, 9)	-0.636429	0.524497
(7, 8)	0.707148	0.479475
(0, 5)	0.721309	0.470721
(3, 7)	-0.77786	0.436653
(2, 8)	0.78729	0.431113
(1, 6)	-0.824977	0.409385
(5, 9)	-0.843874	0.398741
(2, 3)	0.858001	0.390893
(1, 8)	0.895717	0.370405
(2, 6)	-0.933404	0.350612
(1, 3)	0.966428	0.333831
(5, 7)	-0.985305	0.324476
(6, 7)	1.013546	0.310801
(2, 5)	1.065447	0.286675
(6, 9)	1.154977	0.248101
(1, 5)	1.173874	0.240447
(0, 6)	-1.277543	0.201412
(1, 4)	-1.29639	0.194843
(2, 4)	-1.404817	0.160077
(4, 7)	1.484959	0.137556
(4, 9)	1.626391	0.103868
(6, 8)	1.720701	0.085307
(0, 4)	-1.748958	0.080313
(3, 6)	-1.791414	0.073229
(5, 6)	-1.998862	0.045625
(4, 8)	2.19212	0.028372
(3, 4)	-2.262834	0.023647
(4, 5)	2.470284	0.013501

Table 5: Summary of the t-statistics and corresponding p-values for selected task pairs, arranged from most similar (highest p-value) to least similar (lowest p-value)

4.1.3.2 Rotated MNIST

In the Rotated MNIST variant, Table 6 displays the Jaccard index overlap for supermasks across tasks, which is also visualised in a 3D scatter plot, as shown in Figure 25.

Moreover, Figure 26 graphically depicts the individual neural activity of the supermasks for each task within this variant. Table 7 presents a tabulated summary of the t-test results for all task pairs, arranged to highlight pairs from the most to the least similar based on their p-values.

Task ID	**Task ID**	**Overlap**
Task: 0	Task: 1	Overlap: 33.22%
Task: 0	Task: 2	Overlap: 32.85%
Task: 0	Task: 3	Overlap: 33.05%
Task: 0	Task: 4	Overlap: 32.86%
Task: 0	Task: 5	Overlap: 33.32%
Task: 0	Task: 6	Overlap: 33.38%
Task: 0	Task: 7	Overlap: 33.06%
Task: 0	Task: 8	Overlap: 33.11%
Task: 0	Task: 9	Overlap: 33.04%
Task: 1	Task: 0	Overlap: 33.22%
Task: 1	Task: 2	Overlap: 33.03%
Task: 1	Task: 3	Overlap: 33.13%
Task: 1	Task: 4	Overlap: 33.05%
Task: 1	Task: 5	Overlap: 33.35%
Task: 1	Task: 6	Overlap: 33.28%
Task: 1	Task: 7	Overlap: 33.36%
Task: 1	Task: 8	Overlap: 33.12%
Task: 1	Task: 9	Overlap: 33.21%
Task: 2	Task: 0	Overlap: 32.85%
Task: 2	Task: 1	Overlap: 33.03%
Task: 2	Task: 3	Overlap: 32.82%
Task: 2	Task: 4	Overlap: 33.01%
Task: 2	Task: 5	Overlap: 32.97%
Task: 2	Task: 6	Overlap: 32.78%
Task: 2	Task: 7	Overlap: 32.81%
Task: 2	Task: 8	Overlap: 33.05%
Task: 2	Task: 9	Overlap: 32.77%
Task: 3	Task: 0	Overlap: 33.05%
Task: 3	Task: 1	Overlap: 33.14%
Task: 3	Task: 2	Overlap: 32.82%
Task: 3	Task: 4	Overlap: 33.11%
Task: 3	Task: 5	Overlap: 32.99%
Task: 3	Task: 6	Overlap: 32.78%
Task: 3	Task: 7	Overlap: 33.59%
Task: 3	Task: 8	Overlap: 33.29%
Task: 3	Task: 9	Overlap: 33.32%
Task: 4	Task: 0	Overlap: 32.86%
Task: 4	Task: 1	Overlap: 33.05%
Task: 4	Task: 2	Overlap: 33.01%

Task: 4	Task: 3	Overlap: 33.11%
Task: 4	Task: 5	Overlap: 33.22%
Task: 4	Task: 6	Overlap: 33.04%
Task: 4	Task: 7	Overlap: 33.23%
Task: 4	Task: 8	Overlap: 32.98%
Task: 4	Task: 9	Overlap: 32.99%
Task: 5	Task: 0	Overlap: 33.32%
Task: 5	Task: 1	Overlap: 33.35%
Task: 5	Task: 2	Overlap: 32.97%
Task: 5	Task: 3	Overlap: 32.99%
Task: 5	Task: 4	Overlap: 33.22%
Task: 5	Task: 6	Overlap: 33.41%
Task: 5	Task: 7	Overlap: 33.12%
Task: 5	Task: 8	Overlap: 33.08%
Task: 5	Task: 9	Overlap: 32.96%
Task: 6	Task: 0	Overlap: 33.38%
Task: 6	Task: 1	Overlap: 33.28%
Task: 6	Task: 2	Overlap: 32.78%
Task: 6	Task: 3	Overlap: 32.78%
Task: 6	Task: 4	Overlap: 33.04%
Task: 6	Task: 5	Overlap: 33.41%
Task: 6	Task: 7	Overlap: 32.92%
Task: 6	Task: 8	Overlap: 33.15%
Task: 6	Task: 9	Overlap: 32.92%
Task: 7	Task: 0	Overlap: 33.06%
Task: 7	Task: 1	Overlap: 33.36%
Task: 7	Task: 2	Overlap: 32.81%
Task: 7	Task: 3	Overlap: 33.59%
Task: 7	Task: 4	Overlap: 33.23%
Task: 7	Task: 5	Overlap: 33.12%
Task: 7	Task: 6	Overlap: 32.92%
Task: 7	Task: 8	Overlap: 33.18%
Task: 7	Task: 9	Overlap: 33.16%
Task: 8	Task: 0	Overlap: 33.11%
Task: 8	Task: 1	Overlap: 33.12%
Task: 8	Task: 2	Overlap: 33.05%
Task: 8	Task: 3	Overlap: 33.29%
Task: 8	Task: 4	Overlap: 32.98%
Task: 8	Task: 5	Overlap: 33.08%
Task: 8	Task: 6	Overlap: 33.15%
Task: 8	Task: 7	Overlap: 33.18%
Task: 8	Task: 9	Overlap: 32.98%
Task: 9	Task: 0	Overlap: 33.04%
Task: 9	Task: 1	Overlap: 33.24%
Task: 9	Task: 2	Overlap: 32.77%
Task: 9	Task: 3	Overlap: 33.32%
Task: 9	Task: 4	Overlap: 32.99%
Task: 9	Task: 5	Overlap: 32.96%
Task: 9	Task: 6	Overlap: 32.92%
Task: 9	Task: 7	Overlap: 33.16%
Task: 9	Task: 8	Overlap: 32.98%

Table 6: Depiction of the percentage overlap among supermasks for 10 task pairs

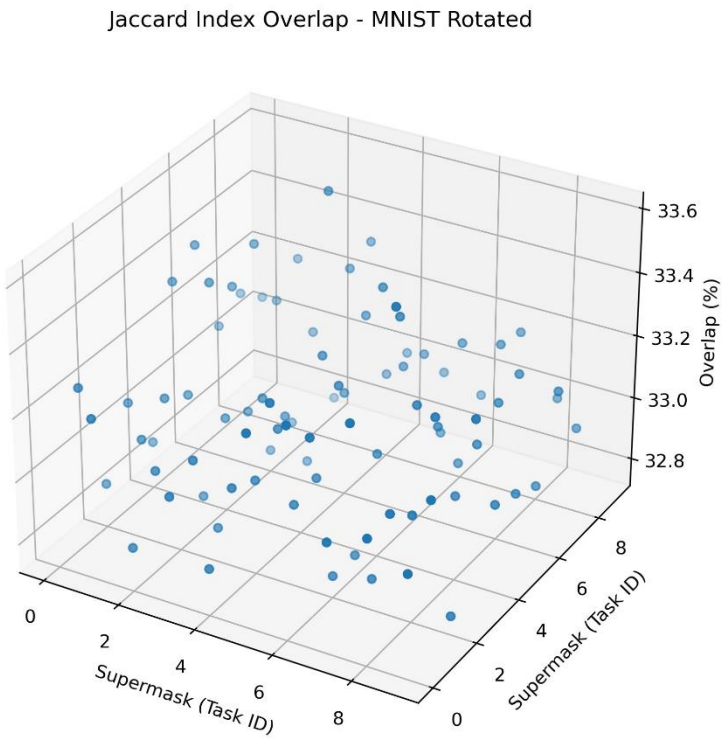

Figure 25: Visual illustration of the percentage overlap among supermasks for 10 task pairs

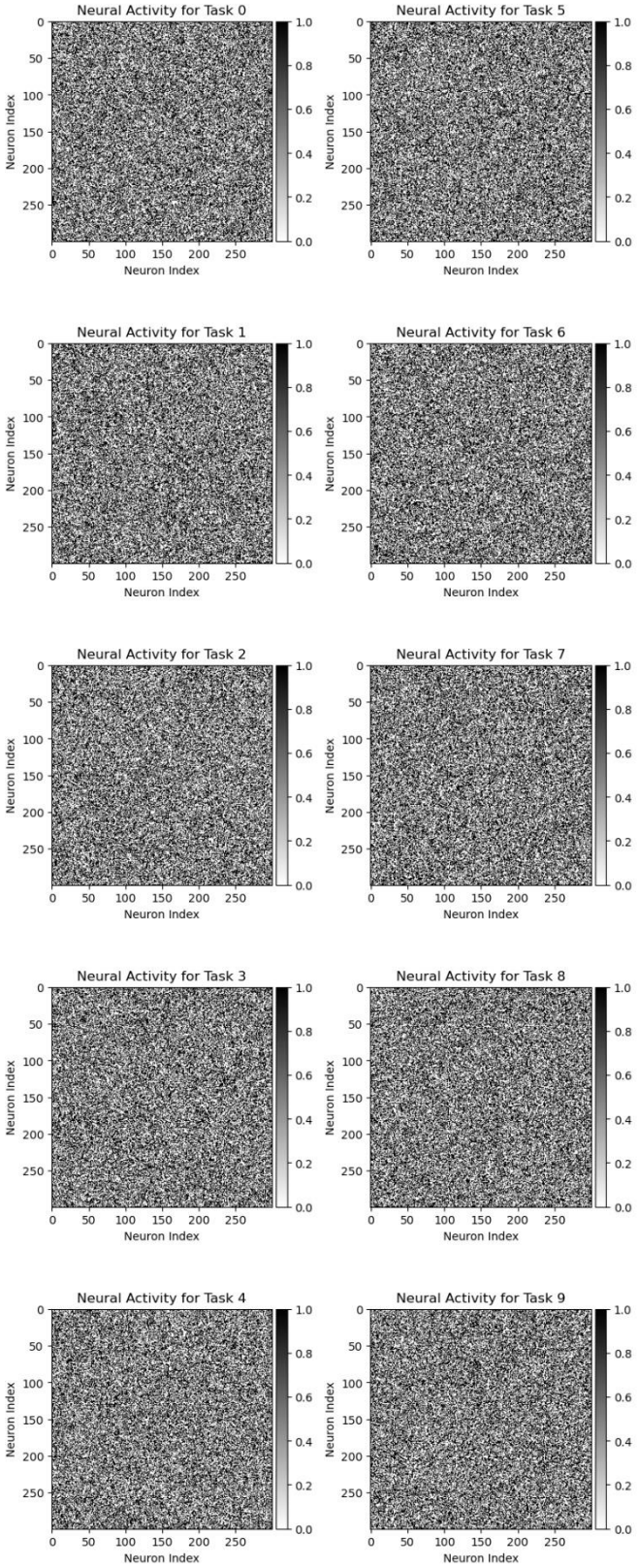

Figure 26: Visual display of neural activity patterns over various tasks. Each subplot signifies neural activity for a distinct task, where darker hues suggest heightened activity

Task Pair	t-statistic	p-value
(6, 8)	0.047138	0.962403
(3, 6)	0.089572	0.928627
(5, 7)	-0.113127	0.909934
(5, 9)	0.117857	0.906181
(3, 8)	0.136711	0.891261
(3, 9)	-0.169711	0.865237
(7, 9)	0.230984	0.817327
(0, 8)	-0.230988	0.817324
(6, 9)	-0.259284	0.795417
(0, 6)	-0.278126	0.780916
(3, 5)	-0.287568	0.773678
(8, 9)	-0.306422	0.759284
(0, 3)	-0.367699	0.713098
(5, 6)	0.377141	0.706072
(3, 7)	-0.400696	0.688645
(5, 8)	0.424279	0.671363
(6, 7)	-0.490268	0.623945
(7, 8)	0.537406	0.590988
(0, 9)	-0.53741	0.590985
(1, 7)	0.612834	0.539987
(0, 5)	-0.655267	0.512296
(1, 5)	0.725961	0.467863
(0, 7)	-0.768395	0.442254
(0, 4)	0.839156	0.401383
(1, 9)	0.843818	0.398772
(1, 3)	1.013531	0.310808
(4, 8)	-1.070145	0.284555
(1, 6)	1.103104	0.269984
(4, 6)	-1.117284	0.263874
(1, 8)	1.150242	0.250046
(3, 4)	1.206857	0.227489
(2, 4)	-1.362533	0.173032
(4, 9)	-1.37657	0.168647
(0, 1)	-1.381232	0.167209
(4, 5)	-1.494428	0.135066
(4, 7)	-1.607557	0.107934
(0, 2)	2.201703	0.027688
(1, 4)	2.220403	0.026393
(2, 8)	-2.432698	0.014988
(2, 6)	-2.479837	0.013145
(2, 3)	-2.569413	0.010188
(2, 9)	-2.739131	0.006161
(2, 5)	-2.856993	0.004277
(2, 7)	-2.970126	0.002977
(1, 2)	3.582995	0.000342

Table 7: Overview of t-statistics and associated p-values for chosen task pairs, ordered from the most similar (with the highest p-value) to the least similar (with the lowest p-value)

4.1.3.3 Partitioned MNIST

For the Partitioned MNIST variant, the Jaccard index overlap for supermasks across tasks is presented in Table 8 and further illustrated in a 3D scatter plot, as seen in Figure 27.

Additionally, the individual neural activity of the supermasks for every task within this variant is portrayed in Figure 28. Table 9 offers a structured overview of the t-test outcomes for all task pairs, organised to emphasise pairs from the most similar to the least, based on their p-values.

Task ID	Task ID	Overlap
Task: 0	Task: 1	Overlap: 33.02%
Task: 0	Task: 2	Overlap: 32.91%
Task: 0	Task: 3	Overlap: 33.21%
Task: 0	Task: 4	Overlap: 33.14%
Task: 0	Task: 5	Overlap: 33.19%
Task: 0	Task: 6	Overlap: 33.21%
Task: 0	Task: 7	Overlap: 33.11%
Task: 0	Task: 8	Overlap: 33.21%
Task: 0	Task: 9	Overlap: 32.89%
Task: 1	Task: 0	Overlap: 33.02%
Task: 1	Task: 2	Overlap: 33.21%
Task: 1	Task: 3	Overlap: 33.37%
Task: 1	Task: 4	Overlap: 33.02%
Task: 1	Task: 5	Overlap: 33.02%
Task: 1	Task: 6	Overlap: 33.23%
Task: 1	Task: 7	Overlap: 33.03%
Task: 1	Task: 8	Overlap: 32.93%
Task: 1	Task: 9	Overlap: 32.86%
Task: 2	Task: 0	Overlap: 32.91%
Task: 2	Task: 1	Overlap: 33.21%
Task: 2	Task: 3	Overlap: 33.09%
Task: 2	Task: 4	Overlap: 33.14%
Task: 2	Task: 5	Overlap: 33.24%
Task: 2	Task: 6	Overlap: 33.18%
Task: 2	Task: 7	Overlap: 33.04%
Task: 2	Task: 8	Overlap: 32.99%
Task: 2	Task: 9	Overlap: 32.83%
Task: 3	Task: 0	Overlap: 33.21%
Task: 3	Task: 1	Overlap: 33.37%
Task: 3	Task: 2	Overlap: 33.09%
Task: 3	Task: 4	Overlap: 32.97%
Task: 3	Task: 5	Overlap: 32.98%
Task: 3	Task: 6	Overlap: 33.05%
Task: 3	Task: 7	Overlap: 33.62%
Task: 3	Task: 8	Overlap: 33.27%
Task: 3	Task: 9	Overlap: 33.43%
Task: 4	Task: 0	Overlap: 33.14%
Task: 4	Task: 1	Overlap: 33.02%
Task: 4	Task: 2	Overlap: 33.14%

Task: 4	Task: 3	Overlap: 32.97%
Task: 4	Task: 5	Overlap: 33.07%
Task: 4	Task: 6	Overlap: 33.01%
Task: 4	Task: 7	Overlap: 32.84%
Task: 4	Task: 8	Overlap: 32.93%
Task: 4	Task: 9	Overlap: 32.93%
Task: 5	Task: 0	Overlap: 33.19%
Task: 5	Task: 1	Overlap: 33.02%
Task: 5	Task: 2	Overlap: 33.24%
Task: 5	Task: 3	Overlap: 32.98%
Task: 5	Task: 4	Overlap: 33.07%
Task: 5	Task: 6	Overlap: 33.42%
Task: 5	Task: 7	Overlap: 33.06%
Task: 5	Task: 8	Overlap: 33.21%
Task: 5	Task: 9	Overlap: 33.06%
Task: 6	Task: 0	Overlap: 33.21%
Task: 6	Task: 1	Overlap: 33.21%
Task: 6	Task: 2	Overlap: 33.18%
Task: 6	Task: 3	Overlap: 33.05%
Task: 6	Task: 4	Overlap: 33.01%
Task: 6	Task: 5	Overlap: 33.42%
Task: 6	Task: 7	Overlap: 33.31%
Task: 6	Task: 8	Overlap: 32.78%
Task: 6	Task: 9	Overlap: 32.95%
Task: 7	Task: 0	Overlap: 33.11%
Task: 7	Task: 1	Overlap: 33.03%
Task: 7	Task: 2	Overlap: 33.02%
Task: 7	Task: 3	Overlap: 33.64%
Task: 7	Task: 4	Overlap: 32.84%
Task: 7	Task: 5	Overlap: 33.06%
Task: 7	Task: 6	Overlap: 33.34%
Task: 7	Task: 8	Overlap: 32.96%
Task: 7	Task: 9	Overlap: 32.96%
Task: 8	Task: 0	Overlap: 33.21%
Task: 8	Task: 1	Overlap: 32.93%
Task: 8	Task: 2	Overlap: 32.99%
Task: 8	Task: 3	Overlap: 33.27%
Task: 8	Task: 4	Overlap: 32.93%
Task: 8	Task: 5	Overlap: 33.21%
Task: 8	Task: 6	Overlap: 32.78%
Task: 8	Task: 7	Overlap: 32.96%
Task: 8	Task: 9	Overlap: 33.34%
Task: 9	Task: 0	Overlap: 32.89%
Task: 9	Task: 1	Overlap: 32.86%
Task: 9	Task: 2	Overlap: 32.83%
Task: 9	Task: 3	Overlap: 33.43%
Task: 9	Task: 4	Overlap: 32.93%
Task: 9	Task: 5	Overlap: 33.06%
Task: 9	Task: 6	Overlap: 32.95%
Task: 9	Task: 7	Overlap: 32.96%
Task: 9	Task: 8	Overlap: 33.34%

Table 8: Illustration of the overlap percentage for supermasks across 10 task pairs

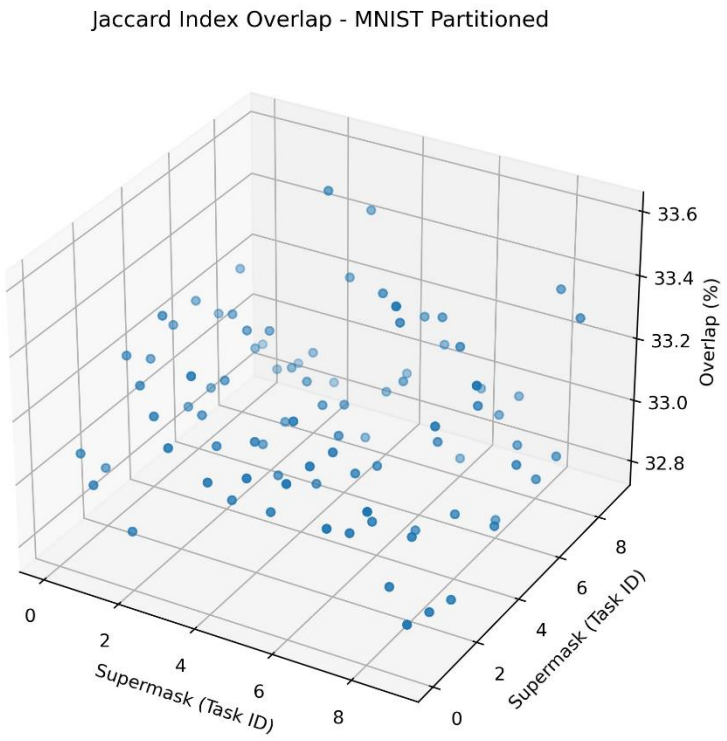

Figure 27: Graphical representation of the overlap percentage for supermasks across 10 task pairs

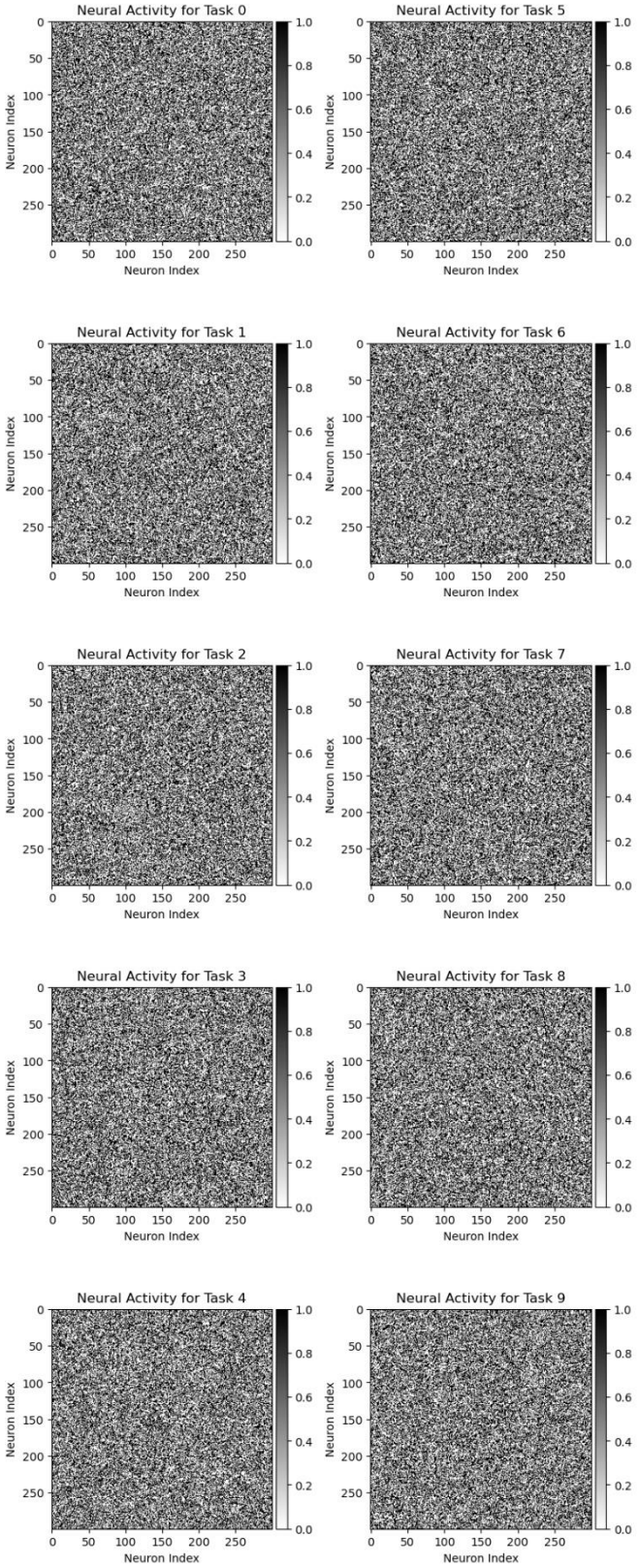

Figure 28: Visual demonstration of neural activity trends across different tasks. Each subplot highlights the neural activity for a particular task, with deeper shades indicating increased activity

Task Pair	t-statistic	p-value
(1, 8)	-0.033002	0.973673
(1, 2)	0.066004	0.947375
(5, 6)	0.09899	0.921146
(2, 8)	-0.099005	0.921134
(0, 6)	-0.146139	0.883811
(0, 5)	-0.245129	0.806357
(0, 7)	0.306403	0.759298
(4, 9)	0.339435	0.734282
(2, 4)	0.419558	0.674809
(6, 7)	0.452543	0.650879
(1, 4)	0.485561	0.627279
(4, 8)	-0.518563	0.604066
(5, 7)	0.551533	0.581269
(3, 5)	0.589248	0.555696
(3, 6)	0.688238	0.491304
(2, 9)	0.758994	0.447857
(1, 9)	0.824997	0.409374
(0, 3)	-0.834378	0.404069
(7, 8)	0.839108	0.401412
(8, 9)	0.857999	0.390894
(1, 7)	-0.87211	0.383155
(2, 7)	-0.938114	0.348187
(3, 7)	1.140783	0.253962
(0, 8)	1.145513	0.251998
(0, 1)	1.178515	0.238593
(0, 2)	1.244519	0.213317
(6, 8)	1.291653	0.196479
(1, 6)	-1.324655	0.185287
(4, 7)	-1.357674	0.174569
(5, 8)	1.390644	0.164335
(2, 6)	-1.39066	0.164334
(1, 5)	-1.423646	0.154551
(2, 5)	-1.489651	0.136318
(0, 4)	1.664081	0.096098
(7, 9)	1.697114	0.089677
(4, 6)	-1.810223	0.070263
(4, 5)	-1.909215	0.056236
(3, 8)	1.979901	0.047716
(0, 9)	2.003523	0.045123
(1, 3)	-2.012904	0.044126
(2, 3)	-2.078909	0.037627
(6, 9)	2.149666	0.031583
(5, 9)	2.248658	0.024535
(3, 4)	2.498479	0.012474
(3, 9)	2.837927	0.004541

Table 9: Summary of t-statistics and their corresponding p-values for selected task pairs, ranked from the most similar (highest p-value) to the least similar (lowest p-value)

4.1.4 Discussion

The results for the Permuted MNIST variant yield overlap percentages between supermasks across various tasks, as illustrated in Table 4. A close observation reveals that the overlaps generally lie in a narrow range between 32.66% to 33.35%. Such minor variations suggest that the supermasks across different tasks for this variant share a significant level of similarity. Visual representations, such as Figure 23, which depicts the overlap percentages in a 3D scatter plot, and Figure 24, which illustrates neural activity patterns, further support these observations. Darker shades in Figure 24 indicate heightened neural activity, which could serve as a cross-reference point to pinpoint specific tasks with increased overlap or similarity. The t-test outcomes presented in Table 5 offer a statistical perspective on the similarity between task pairs. Notably, the highest p-value (0.943628 for the task pair (3, 8)) suggests the most similarity, while the lowest p-value (0.013501 for the task pair (4, 5)) indicates the least. It is paramount to note that p-values above 0.05 generally indicate a lack of significant difference between the groups being compared. In this context, the majority of the task pairs in the Permuted MNIST variant are not significantly different from each other in terms of their overlap.

In the Rotated MNIST variant, the overlap percentages, as shown in Table 6, typically range between 32.77% to 33.59%. This slightly wider range compared to the Permuted MNIST suggests more variation between supermasks across tasks for this variant. Figure 25 offers a 3D scatter plot representation of these overlaps, while Figure 26 showcases the neural activity patterns across various tasks. As with the previous variant, darker hues in Figure 26 are indicative of heightened neural activity. The t-test results in Table 7 reveal the highest p-value of 0.962403 for the task pair (6, 8), marking it as the most similar. The least similarity is observed in the task pair (1, 2) with a p-value of 0.000342. As before, the majority of task pairs exhibit a lack of significant difference in their overlap, further reinforcing the similarity in supermask overlaps across tasks.

For the Partitioned MNIST variant, overlap percentages are primarily observed to fall between 32.78% to 33.64%, as seen in Table 8. This range, similar to the Rotated MNIST variant, points to a reasonable level of similarity between supermasks across tasks. Visual insights from Figure 27, the 3D scatter plot of overlap percentages, and Figure 28, which portrays neural activity patterns, further validate these findings. In Figure 28, deeper shades suggest increased neural activity, which can be cross-referenced with overlap percentages to deduce patterns or outliers. The t-test results, as shown in Table 9, present a highest p-value of 0.973673 for the task pair (1, 8), identifying it as the most similar. In contrast, the task pair (3, 9) with a p-value of 0.004541 is the least similar. Again, a significant portion of the task pairs in this variant do not differ significantly in their overlap, underscoring the recurring theme of notable similarity in supermask overlaps across tasks in all variants.

Across all three MNIST variants, there is a substantial overlap in supermasks across different tasks. The narrow range of overlap percentages and the high p-values from the t-tests further substantiate this observation. Such findings are central for understanding the neural network's task-agnostic capabilities and could have implications for future research on network pruning, task learning efficiency, and model generalisation.

4.2 Experiment 2

4.2.1 Overview

In this experiment, we delve into our two distinct, classic and novel, approaches. While the classic approach offers a foundational methodology of training models across different MNIST variations, the novel approach introduces an innovative dimension by incorporating task similarities in the learning process.

The core hypothesis underpinning this novel approach is to determine whether the incorporation of task similarities can lead to enhanced learning efficiency. By comparing these methods, we aim to discern the efficacy and nuances of each, validating or refuting our hypothesis.

4.2.2 Experimental Procedures

4.2.2.1 Foundational Approach (Classic)

For the classic approach, we devised a method to systematically train models across different tasks using the previously mentioned distinct variations of the MNIST dataset. This was achieved through the training_classic_ev function. The primary objective of this function is to train models on three variations of the MNIST dataset. For each of these datasets, the model undergoes training for 10 training rounds.

The function signature for the classic approach, training_classic_ev, encompasses three arguments: the number of training epochs which is set to 1, the total number of tasks which is set to 10, and the size of the hidden layer which is set to 300. Upon completion of the training across all tasks and datasets, the function returns a comprehensive dictionary.

This dictionary captures the accuracy, training time and loss of the model for each task and dataset variation.

Commencing with the permuted MNIST dataset as an illustration, the training procedure involves several systematic steps. Initially, the permuted MNIST dataset is loaded into the function using the MNISTPerm class. Concurrently, the classic multitask model, MultitaskFC, is initialised based on the set hidden size and number of tasks. As we delve into the training loop, for each task, the current task is set both in the model and the dataset. The RMSprop optimiser, with a learning rate of 1×10^{-4}, is subsequently initialised.

During each epoch, the model undergoes training on the dataset with metrics such as batch normalisation means being calculated. Following the training, the model's performance is assessed on a validation dataset. Post-training for each task, the model's state, and performance metrics such as accuracy, training time, and loss are stored. Additionally, the model's state after each task's training is saved for future utilisation in the novel approach.

A distinguishing feature of our classic approach is the calculation of task similarities post-training using cosine similarity. Leveraging the batch normalisation means acquired during training, the calculate_task_similarityE1 function computes a task similarity matrix. This matrix is then visually represented as a heatmap, providing an intuitive understanding of the overlap between masks for different tasks based on task similarity.

Concluding the training process for the permuted MNIST dataset, average performance metrics across all tasks are calculated. Specifically, we derive the average accuracy, training time, and loss which are then incorporated into the aforementioned results dictionary. The described procedure is reiterated for the other two MNIST dataset variations.

4.2.2.2 Incorporating Task Similarity Approach (Novel)

The training_novel_ev function represents our novel approach, specifically tailored for tasks using different versions of the MNIST dataset. The core hypothesis underpinning this approach is to determine whether the incorporation of task similarities can lead to enhanced learning efficiency.

Similarly, the function is designed to train on the three distinct variations of the MNIST dataset: permuted, rotated, and partitioned. For each variation, the model undergoes training across the 10 tasks for 10 training rounds. Similar to the classic approach parameters

for the function include the number of epochs which is set to 1, the total number of tasks which is set to 10, and the hidden layer's dimensionality which is set to 300. In addition, the RMSprop optimiser, with a learning rate of 1×10^{-4}, is subsequently initialised.

Initially, the permuted MNIST dataset is loaded, and a specific model, MultitaskFCV2, is initialised. One key distinction in this approach is the integration of batch normalisation means values, extracted from previously trained and saved classic approach models. These values provide insights into activation distributions and are critical for determining task similarities.

During training, the batch normalisation means values are used to compute a task similarity matrix. This matrix aids in the determination and setting of 'alphas'—the aforementioned coefficients that modulate the influence of a preceding most similar task on the current one. It is postulated that by utilising this influence based on the intrinsic similarities between the tasks, the learning process can be rendered more efficient.

The training process is replicated for the rotated and partitioned MNIST datasets. For each dataset variant, the function captures performance metrics, particularly, accuracy, training time, and loss for each task. Upon completion, the function returns a comprehensive results dictionary, detailing performance across all tasks and datasets.

4.2.3 Experimental Results

4.2.3.1 Task Similarity Calculation (Classic)

For the classic approach, task similarities were computed using cosine similarity utilising the batch normalisation means extracted during training. These similarities were visualised as heatmaps to help in discerning the overlap between the supermasks for different tasks and providing insights into how distinct or similar certain tasks are in relation to each other. The heatmaps for each MNIST dataset variation (permuted, rotated, and partitioned) are represented in Figures 29, 30 and 31, respectively.

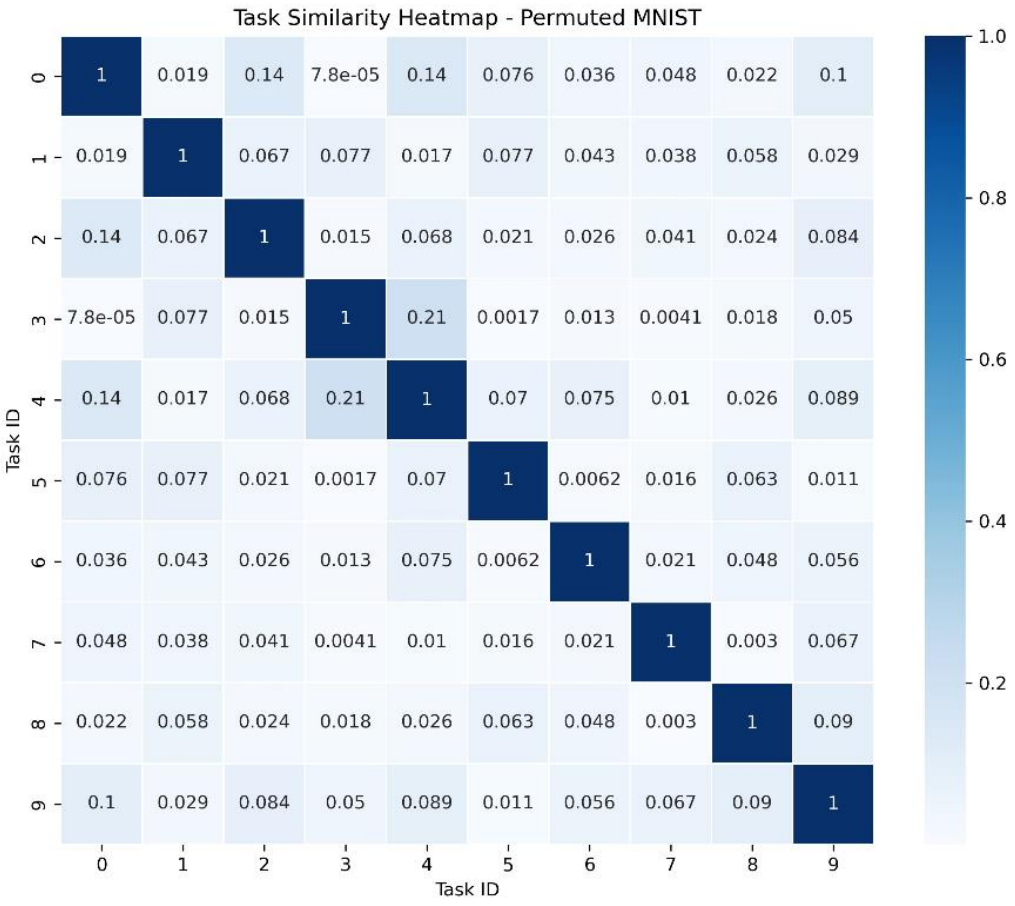

Figure 29: Visual representation of task-related similarities within the Permutated MNIST dataset

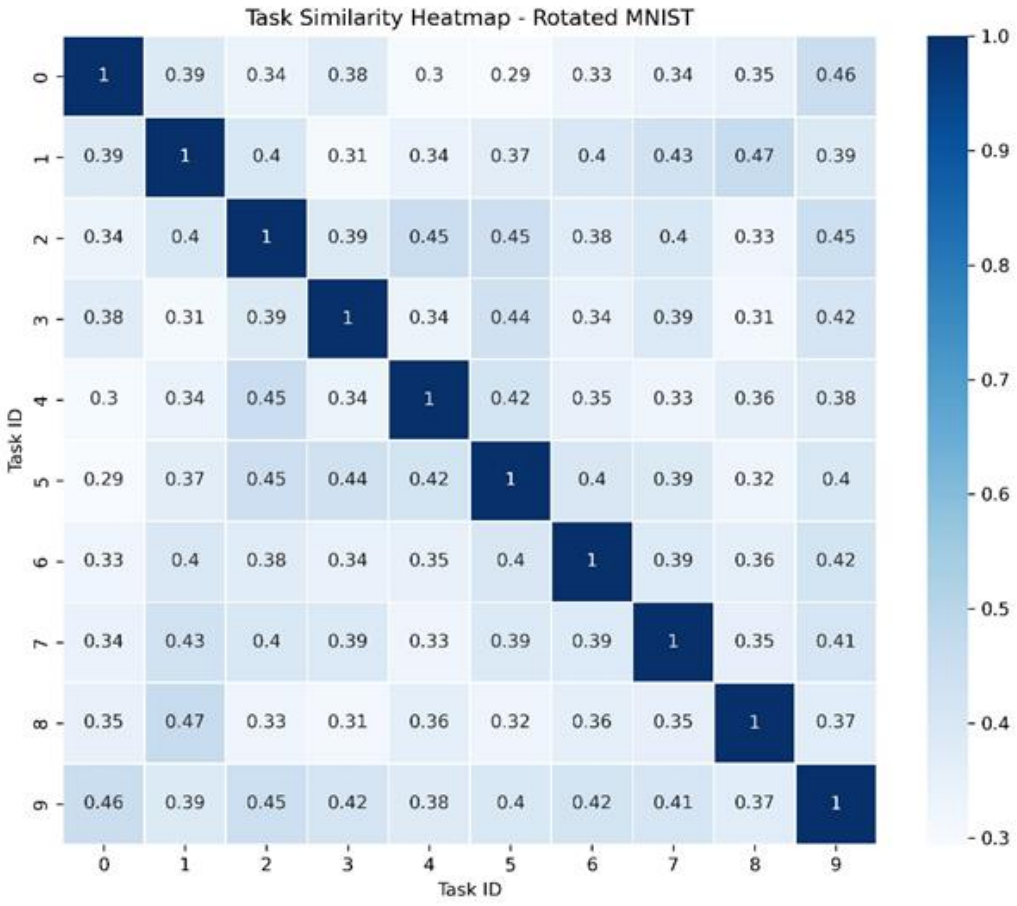

Figure 30: Visual representation of task-related similarities within the Rotated MNIST dataset

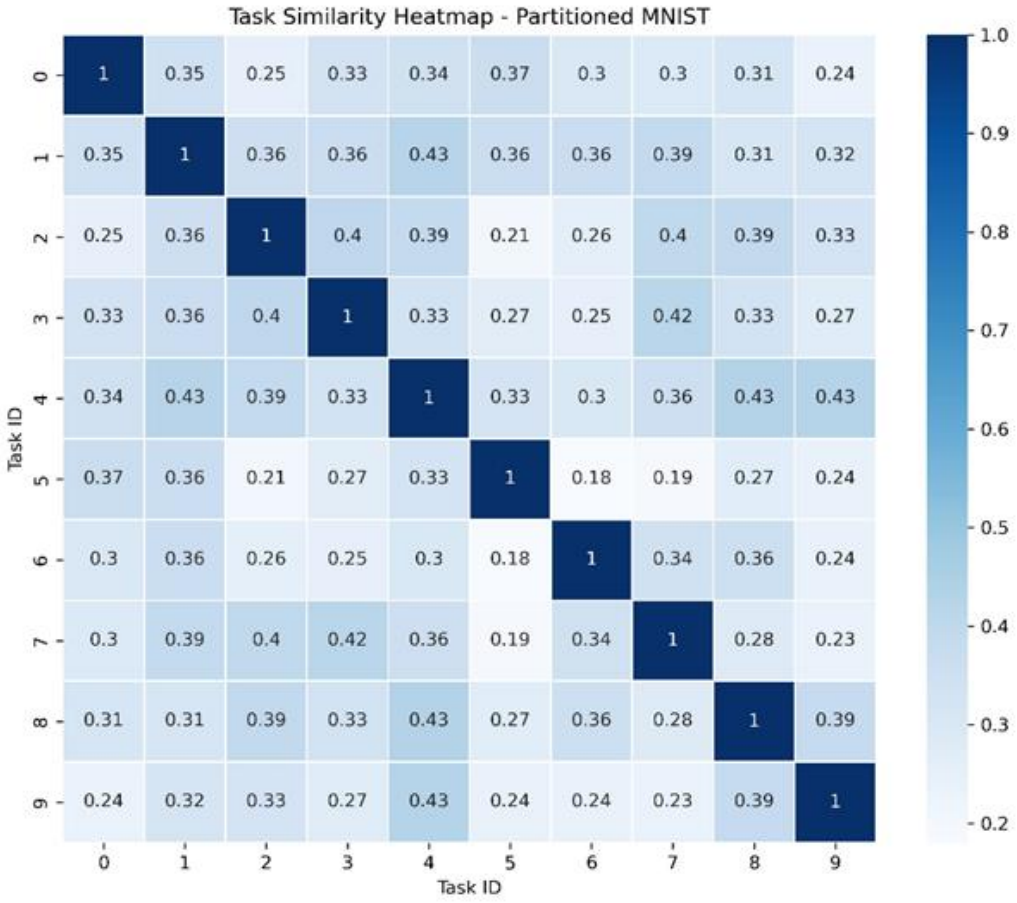

Figure 31: Visual representation of task-related similarities within the Partitioned MNIST dataset

4.2.3.2 Comparative Results of the Classic and Novel Approaches

In our endeavour to provide a comprehensive comparative analysis between the classic and novel approaches, we evaluated the methods across the three distinct dataset variations where each variation was assessed based on the previously mentioned three primary metrics: accuracy, training time, and loss. The following tables illustrate the average performance metrics across the three dataset variations for both the classic and novel approaches:

Metric	Classic Approach	Novel Approach
Accuracy	94.04	94.13
Training Time	88.56 seconds	69.48 seconds
Loss	0.5445	0.5329

Table 10: Comparative average performance metrics of the classic and novel approaches for the Permuted MNIST variant

Metric	Classic Approach	Novel Approach
Accuracy	94.04	94.10
Training Time	95.65 seconds	78.37 seconds
Loss	0.5425	0.5359

Table 11: Comparative average performance metrics of the classic and novel approaches for the Rotated MNIST variant

Metric	Classic Approach	Novel Approach
Accuracy	98.40	98.47
Training Time	15.74 seconds	13.14 seconds
Loss	0.9437	0.9150

Table 12: Comparative performance metrics of the classic and novel approaches for the Partitioned MNIST variant

For the permuted dataset (Table 10), the classic approach yielded an accuracy of 94.04%, a training time of 88.56 seconds, and a loss value of 0.5445. In contrast, the novel approach demonstrated an accuracy of 94.13%, a training time reduced to 69.48 seconds, and a slightly improved loss value of 0.5329. Turning our attention to the rotated dataset (Table 11), the classic approach's performance metrics were as follows: an accuracy of 94.04%, training time of 95.65 seconds, and a loss of 0.5425. The novel approach, on the other hand, achieved an accuracy of 94.10%, reduced the training time to 78.37 seconds, and posted a loss value of 0.5359. Lastly, for the partitioned dataset (Table 12), the classic approach registered an accuracy of 98.40%, a training time of 15.74 seconds, and a loss value of 0.9437. The novel approach slightly outperformed with an accuracy of 98.47%, a slightly diminished training time of 13.14 seconds, and a better loss value of 0.9150. Furthermore, to provide a more vivid and tangible understanding of these metrics, we have also visualised the results, which are presented below.

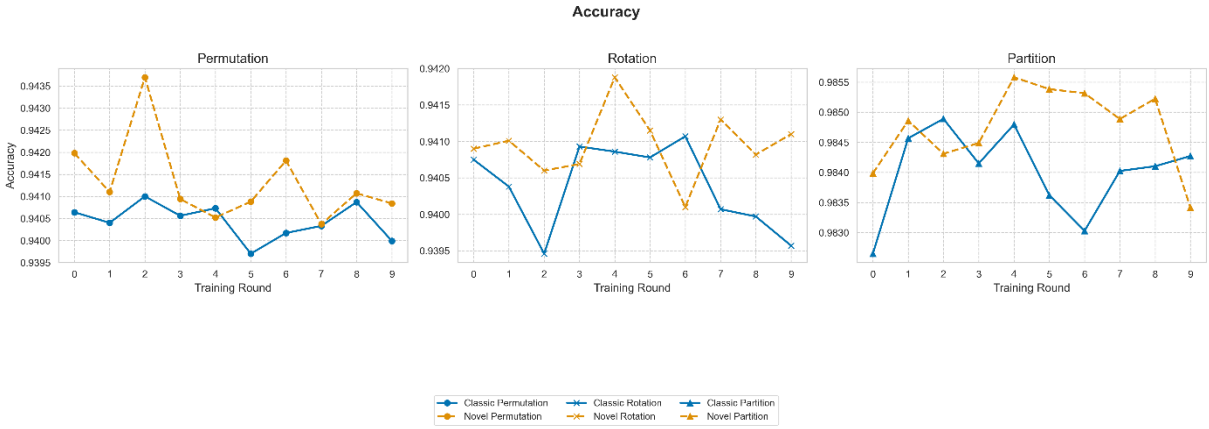

Figure 32: Comparative visualisation of accuracy between the classic and novel Approaches across the three MNIST dataset variants

Figure 33: Comparative visualisation of training time between the Classic and Novel Approaches across the three MNIST dataset variants

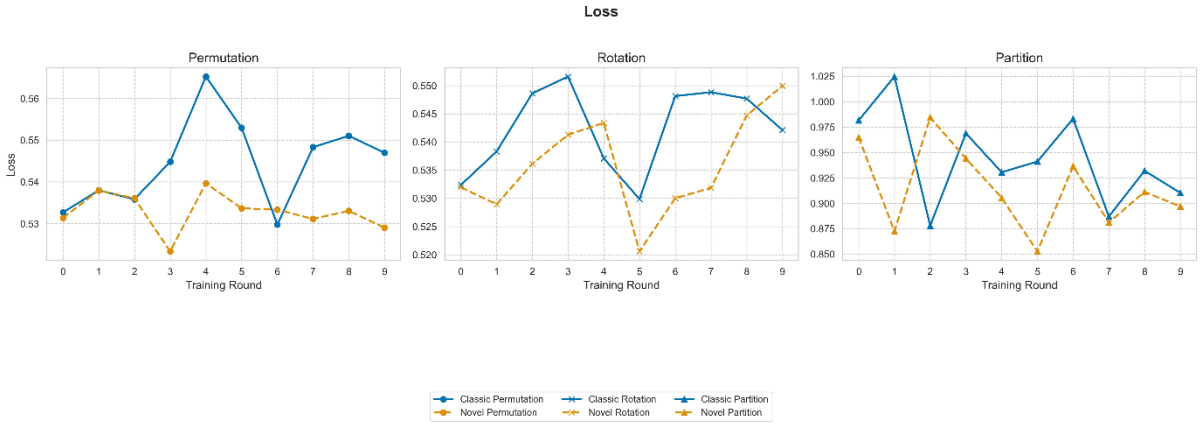

Figure 34: Comparative visualisation of loss values between the Classic and Novel Approaches across the three MNIST dataset variants

4.2.4 Discussion

From our experiment, a few key insights emerge. Firstly, the accuracy of the novel approach is either marginally superior or on par with the classic approach across all dataset variations. Secondly, the training time observed in the novel approach is invariably shorter than its classic counterpart, signalling a heightened efficiency. Lastly, the loss metrics from the novel approach are generally lower or commensurate with the classic approach for all dataset variations. This suggests that the integration of task similarities in the novel approach potentially augments the learning efficiency without detriment to the model's performance.

The central focus of this experiment, in particular, and our research, in general, was to determine the extent of overlaps through task similarity and subsequently ascertain if the utilisation of the supermasks overlap through task similarities into our novel model would improve learning efficiency. The results, derived from both the classic and novel approaches, provide insights into this query.

The results of our experiment reveals that the novel approach consistently requires less training time across all dataset variations compared to the classic method. This reduction in training time suggests a more efficient learning process when task similarities are integrated.

While training efficiency is important, it should not detract from model performance. The accuracy metrics of the novel approach, relative to the classic one, indicate that the efficiency gains do not compromise the quality of learning.

Furthermore, lower loss values in the novel approach, combined with its reduced training times, suggest better model robustness. This implies that the model might be more adept at generalising to new, unseen data.

Based on the results, it's evident that the incorporation and utilisation of task similarities enhances learning efficiency without sacrificing performance. The integration of task similarities emerges to provide the model with a more structured learning pathway. By recognising supermask overlaps or interrelated task relationships, the model seems to optimise its learning process more effectively.

4.3 Experiment 3

4.3.1 Overview

This experiment mirrors the procedures of Experiment 2, focusing on both the classic and novel approaches. While retaining the principles of the foundational methodology, Experiment 3 delves deeper into the evaluation metrics, introducing precision, recall, and the F-1 score as performance indicators. These metrics are indispensable in understanding the nuanced performance of models across different tasks and variations of the MNIST dataset.

The primary objective here is to ascertain if the inclusion of these advanced performance metrics, along with a modified training iterations (set to 2 epochs), provides a more granular and comparative insight into the learning capabilities of our models, while still examining the potential advantages of incorporating task similarities.

4.3.2 Experimental Procedures

4.3.2.1 Foundational Approach (Classic)

The foundational approach for Experiment 3 utilises the training_classic_pd function, with a heightened focus on precision, recall, and F-1 score metrics. The model undergoes training on the MNIST dataset variations: permuted, rotated, and partitioned, executing over a single training round for each.

The function signature for training_classic_pd has been modified for this experiment. The arguments now include a setting for the number of training epochs to 2, while the total number of tasks remains at 10, and the hidden layer size at 300. At the end of the training iterations, validation information was displayed, and predictions were made using the predict function. The predictions and labels were saved in the results dictionary, and the model's state was cached. This dictionary captures the predictions and labels for each task and dataset variation.

The training process, starting with the permuted MNIST dataset, remains consistent with the previous experiment. However, post-training for each task, along with saving the model's state, the performance metrics have been configured to include precision, recall, and

F-1 score. On concluding the training for each MNIST dataset variation, the average values for these new metrics are calculated.

4.3.2.2 Incorporating Task Similarity Approach (Novel)

The training_novel_pd function, in line with the novel approach, is employed for Experiment 3. Its focus is shifted to precision, recall, and F-1 score as primary performance metrics, aiming to further understand the impact of task similarities on model performance.

Training parameters and procedures largely mirror those of Experiment 2. Yet, there are subtle changes. The number of epochs for this function has been increased to 2. And while the RMSprop optimiser remains constant with a learning rate of 1×10^{-4}, the evaluation process post-training now records the predictions and labels to calculate the precision, recall, and F-1 score.

In the context of the permuted MNIST dataset, the entire training process aligns with the previous experiment, with the only variation being the inclusion of the new performance metrics. This ensures a more comprehensive understanding of the model's capability and the efficacy of utilising task similarities in the learning process.

After finishing the training for all MNIST dataset variants, the function returns a more detailed results dictionary, highlighting the predictions and labels for all tasks and dataset variants.

4.3.3 Experimental Results

4.3.3.1 Permuted MNIST

For the Permuted dataset, the confusion matrices illustrating the alignment of predicted and actual classifications for the classic and novel approaches are depicted in Figures 35 and 36, respectively. Moreover, performance metrics detailing precision, recall, F1-score, as well as accuracy across tasks are presented in Tables 13 through 15.

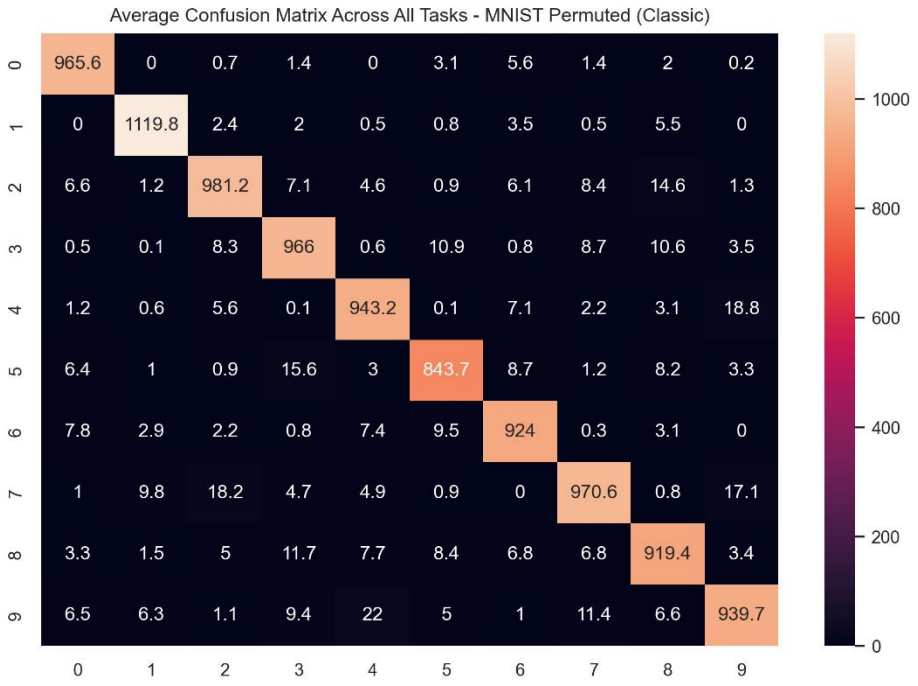

Figure 35: Visual representation of the average confusion matrix across all tasks using the classic approach

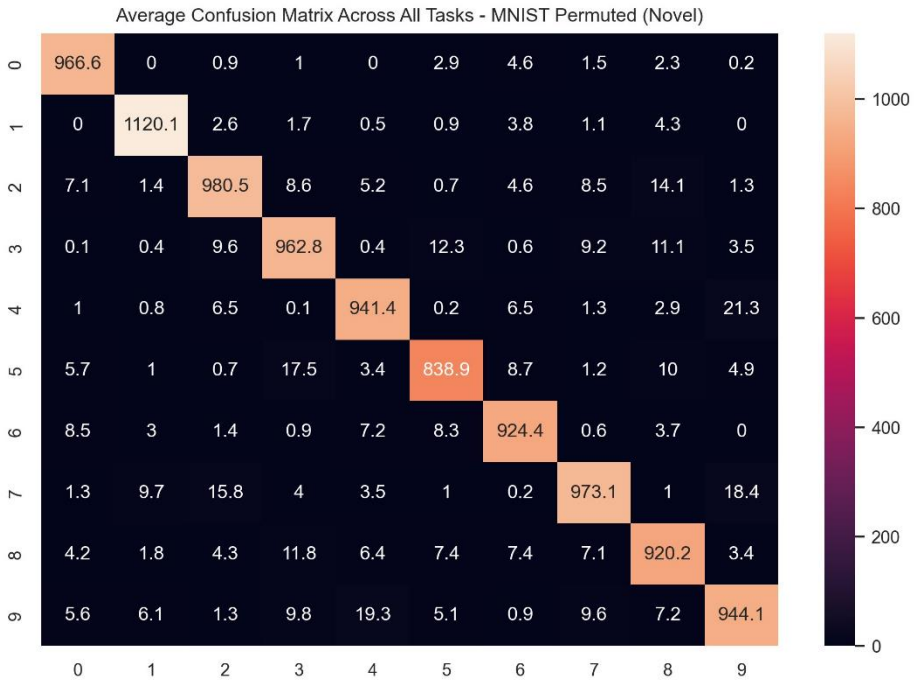

Figure 36: Visual representation of the average confusion matrix across all tasks using the novel approach

Task ID	Precision	Recall	F1-score	Accuracy
0	95.47	95.47	95.46	95.50
1	95.50	95.50	95.50	95.54
2	95.94	95.94	95.93	95.97
3	95.66	95.64	95.64	95.68
4	95.70	95.70	95.69	95.74
5	95.77	95.74	95.75	95.78
6	95.65	95.66	95.65	95.70
7	95.98	95.96	95.97	96.00
8	95.52	95.53	95.52	95.56
9	95.84	95.80	95.81	95.85

Table 13: Overview of the precision, recall, and F1-score metrics for each task using the classic approach

Task ID	Precision	Recall	F1-score	Accuracy
0	95.41	95.39	95.40	95.44
1	95.40	95.40	95.40	95.45
2	95.73	95.72	95.72	95.76
3	95.71	95.68	95.69	95.73
4	95.77	95.76	95.76	95.80
5	95.88	95.87	95.87	95.91
6	95.85	95.83	95.84	95.88
7	95.91	95.86	95.88	95.92
8	95.69	95.70	95.68	95.73
9	95.58	95.54	95.55	95.59

Table 14: Overview of the precision, recall, and F1-score metrics for each task using the novel approach

Metric	Classic Approach	Novel Approach
Precision	95.70	95.69
Recall	95.69	95.68
F1-score	95.69	95.68
Accuracy	95.73	95.72

Table 15: Comparative average performance metrics of the classic and novel approaches

4.3.3.2 Rotated MNIST

Within the Rotated dataset results, Figures 37 and 38 provide visual representation through confusion matrices. Complementing this, Tables 16 through 18 offer a comprehensive breakdown of precision, recall, F1-score, in addition to accuracy, showcasing the models' proficiency on this dataset variant.

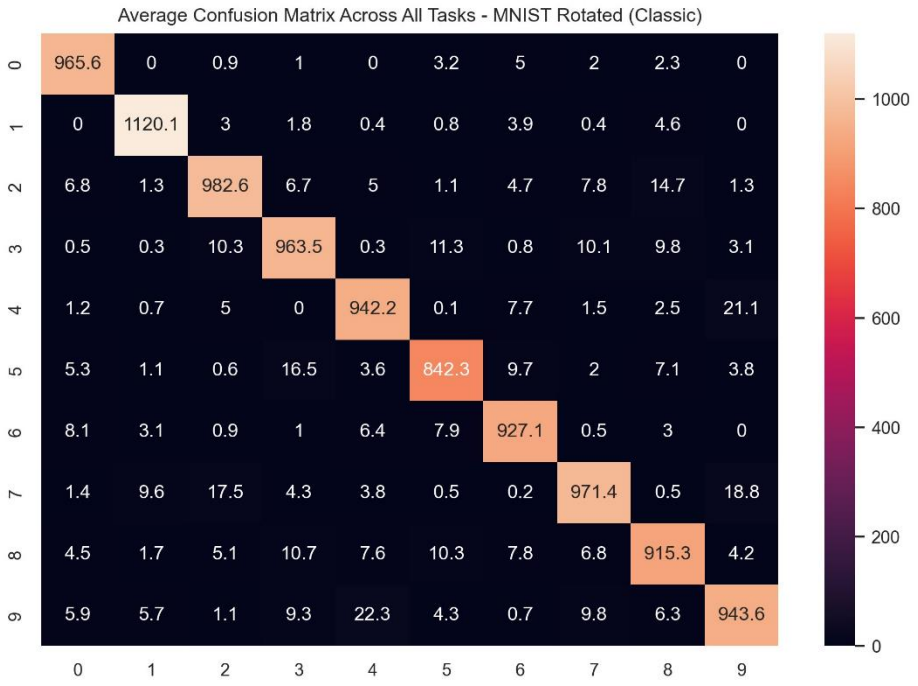

Figure 37: Visual representation of the average confusion matrix across all tasks using the classic approach

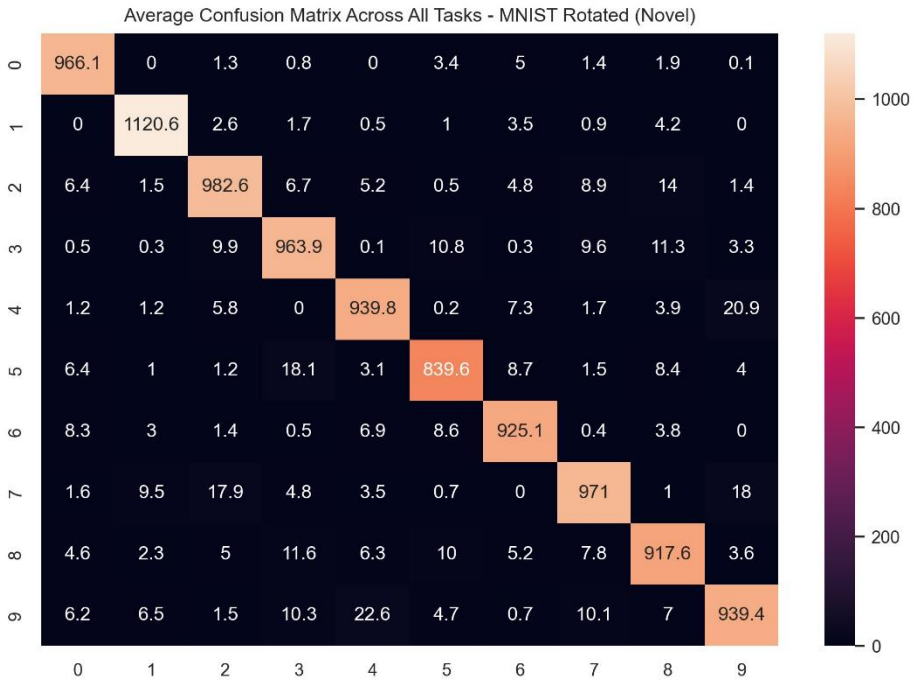

Figure 38: Visual representation of the average confusion matrix across all tasks using the novel approach

Task ID	Precision	Recall	F1-score	Accuracy
0	95.51	95.49	95.50	95.54
1	95.57	95.55	95.56	95.60
2	95.66	95.66	95.65	95.69
3	95.52	95.51	95.51	95.56
4	95.97	95.97	95.97	96.00
5	95.58	95.55	95.55	95.60
6	95.50	95.48	95.48	95.53
7	95.88	95.88	95.87	95.91
8	95.94	95.96	95.95	95.99
9	95.92	95.92	95.91	95.95

Table 16: Overview of the precision, recall, and F1-score metrics for each task using the classic approach

Task ID	Precision	Recall	F1-score	Accuracy
0	95.73	95.71	95.71	95.75
1	95.66	95.64	95.64	95.68
2	95.57	95.56	95.56	95.61
3	95.54	95.51	95.52	95.57
4	95.60	95.58	95.58	95.62
5	95.55	95.52	95.53	95.57
6	95.66	95.64	95.64	95.69
7	95.88	95.86	95.86	95.90
8	95.53	95.52	95.52	95.56
9	95.57	95.58	95.57	95.62

Table 17: Overview of the precision, recall, and F1-score metrics for each task using the novel approach

Metric	Classic Approach	Novel Approach
Precision	95.71	95.63
Recall	95.70	95.61
F1-score	95.70	95.61
Accuracy	95.74	95.66

Table 18: Comparative average performance metrics of the classic and novel approaches

4.3.3.3 Partitioned MNIST

For the Partitioned dataset, a 5x2 grid of confusion matrices is exhibited in Figures 39 and 40, elucidating the model's performance across individual tasks. Additionally, Tables 19 through 21 provide a detailed view into precision, recall, F1-score, in addition to accuracy, providing a comprehensive perspective of the model's capabilities on segmented data.

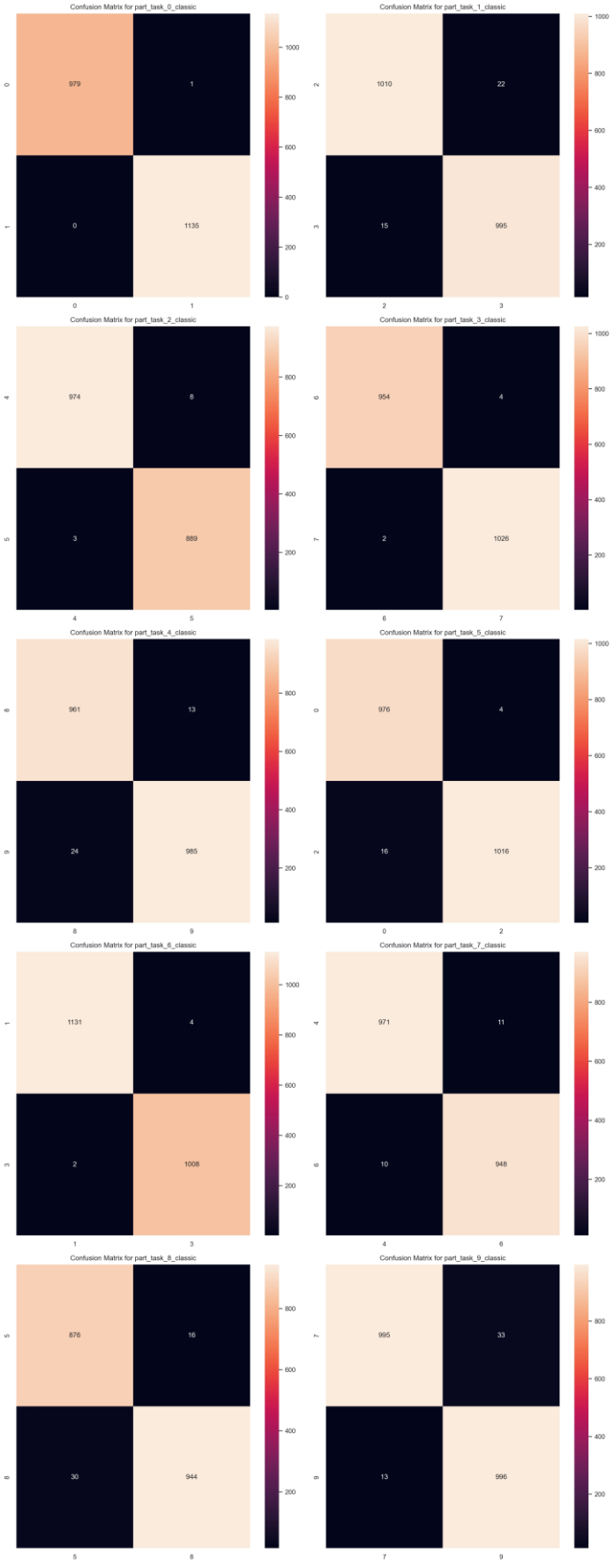

Figure 39: A grid of subplots showcasing the confusion matrices for each task using the classic approach

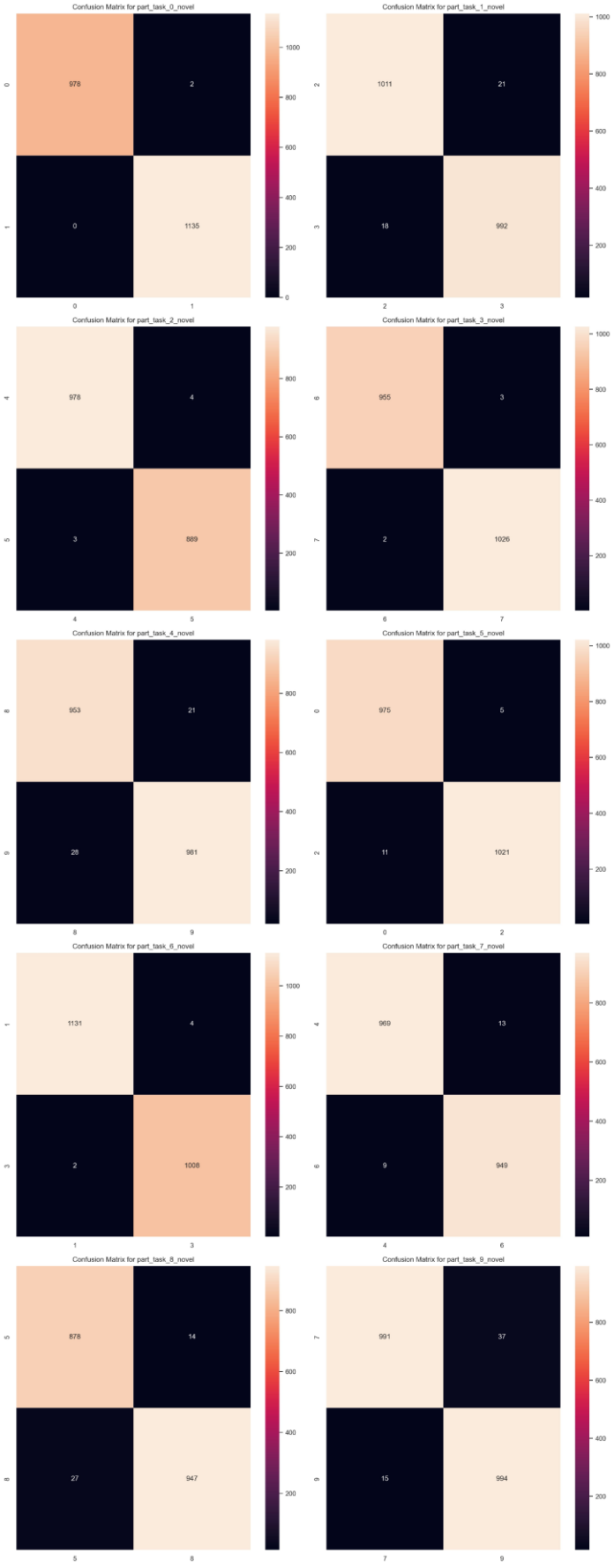

Figure 40: A grid of subplots showcasing the confusion matrices for each task using the novel approach

Task ID	Precision	Recall	F1-score	Accuracy
0	99.96	99.95	99.95	99.95
1	98.19	98.19	98.19	98.19
2	99.40	99.42	99.41	99.41
3	99.70	99.69	99.70	99.70
4	98.13	98.14	98.13	98.13
5	99.00	99.02	99.01	99.01
6	99.71	99.72	99.72	99.72
7	98.92	98.92	98.92	98.92
8	97.51	97.56	97.53	97.53
9	97.75	97.75	97.74	97.74

Table 19: Overview of the precision, recall, and F1-score metrics for each task using the classic approach

Task ID	Precision	Recall	F1-score	Accuracy
0	99.91	99.90	99.90	99.91
1	98.09	98.09	98.09	98.09
2	99.62	99.63	99.63	99.63
3	99.75	99.75	99.75	99.75
4	97.52	97.53	97.53	97.53
5	99.20	99.21	99.20	99.20
6	99.71	99.72	99.72	99.72
7	98.86	98.87	98.87	98.87
8	97.78	97.83	97.80	97.80
9	97.46	97.46	97.45	97.45

Table 20: Overview of the precision, recall, and F1-score metrics for each task using the novel approach

Metric	Classic Approach	Novel Approach
Precision	98.83	98.79
Recall	98.84	98.80
F1-score	98.83	98.79
Accuracy	98.83	98.80

Table 21: Comparative average performance metrics of the classic and novel approaches

4.3.4 Discussion

In Experiment 3, the inclusion of advanced performance metrics—precision, recall, and the F-1 score—was aimed at providing a more comprehensive evaluation of both the classic and novel approaches across different variations of the MNIST dataset.

Upon examination of the Permuted MNIST results, it is evident that the performance metrics for both the classic and novel approaches are closely aligned. The precision, recall, and F1-score metrics for both methods gravitate around the mid-95% range (as observed in Tables 13, 14, and 15). Such proximate results suggest a high level of proficiency in distinguishing between digit classes post-permutation. Notably, the classic approach demonstrates a slightly superior average across metrics, although the difference is marginal.

This subtle distinction may underscore the classic approach's capability to generalise minutely better on permuted datasets.

For the Rotated MNIST dataset, the classic approach exhibited a consistent yet slight outperformance compared to the novel approach. This is evident in the differential depicted in Table 18. One could postulate that the inherent complexities introduced by rotation might render the task similarities (inherent in the novel approach) less effective. The disparities, although minor, might be indicative of the challenges posed by rotational variations on the dataset and how each approach addresses them.

The results from the Partitioned MNIST dataset are particularly illuminating. Due to the inherent nature of a partitioned dataset—wherein only a subset of classes is present in each task—elevated performance metrics were anticipated. Both methodologies yielded exceptional results, with the classic approach marginally surpassing the novel approach in terms of precision, recall, and F1-score (as detailed in Table 21). Such findings could suggest that, in the context of partitioned data, the advantages of task similarities may be somewhat diminished. However, the slight variances between the two approaches further emphasise their respective robustness.

In synthesising the results across the three datasets, it becomes evident that both the classic and novel methodologies are highly competent, with neither conclusively outperforming the other across all variations. The nuanced differences underscore the intricate dynamics of each dataset variation and how each approach navigates them.

4.4 Experiment 2 and Experiment 3 Comparative Discussion

The results from Experiments 2 and 3 present a detailed perspective on the performance dynamics between the classic and novel approaches when applied to the MNIST dataset and its variations. Experiment 2 primarily focused on the influence of task similarities on accuracy, training time and loss. The results indicated that the novel approach consistently required less training time across all dataset variations, suggesting enhanced learning efficiency. Moreover, in terms of accuracy and loss metrics, the novel method often marginally outperformed or matched the classic approach's performance. This was a promising indication that the integration of supermask overlaps through task similarities could potentially enhance learning efficiency without compromising on performance.

However, Experiment 3, while building on the foundation of Experiment 2, introduced a different set of evaluation metrics—precision, recall, and F1-score—and adjusted the training iterations. With this nuanced methodology, the comparative dynamics between the two approaches shifted. The classic approach demonstrated a slight advantage in performance across the three dataset variants.

Such a divergence in performance between the two experiments is instructive. It points to the sensitivity of machine learning models to evaluation metrics and the modifications in the experimental procedures employed. While the novel approach's emphasis on task similarities showed clear advantages in terms of training efficiency, and marginal advantages in performance, in Experiment 2, its benefits became less distinct when subjected to epoch modifications and the granular evaluation metrics of Experiment 3. This suggests that, although task similarities can enhance certain aspects of model performance, they may not universally provide advantages across scenarios.

While both approaches have demonstrated merits in specific contexts, continuous evaluation and adaptation are essential in machine learning research. The observed dynamics between the two approaches across these experiments underline the importance of iterative refinement and re-evaluation. It would be beneficial for future research to further refine the novel approach, exploring how task similarities can be more effectively leveraged, and to consider other methodologies that might enhance performance across a broader spectrum of metrics and datasets.

4.5 Conclusion

The series of experiments conducted provide intriguing insights into the dynamics of representation sharing and its potential to enhance learning efficiency across varied permutations of the foundational MNIST dataset. While affirming existing principles, they also reveal nuanced limitations that warrant further investigation.

In Experiment 1, we quantified and established that, despite input transformations, neural networks leverage substantial representational overlap across MNIST tasks. The consistently high supermask overlaps and statistically insignificant differences between most task pairs underline reused features facilitating generalisation. However, the extent of this overlap varies subtly with the nature of the transformations, exposing the network's sensitivity.

Furthermore, Experiment 2 offered a promising outlook on integrating task similarities through supermask overlaps to improve learning efficiency. The experiment results revealed that the novel approach consistently required less training time compared to the classic approach, affirming the utilisation of supermask overlaps to achieve accelerated learning with uncompromised performance. Yet, Experiment 3 provided a critical assessment, demonstrating the classic approach's slight effectiveness across training procedure modifications and more granular metrics. This divergence emphasised that, while task similarities provide measurable advantages, their benefits are not universal.

Synthesising the insights, it is evident that supermask overlaps are a pivotal asset for generalisation in neural networks, with its advantages contingent on the methodologies and datasets involved. While the novel approach shows clear promise in enhancing efficiency, further research is imperative to fully unlock the potential of task similarities across diverse contexts. The intricacies exposed warrant meticulous examination through rigorous statistical analyses paired with more adaptable techniques.

The experiments provided a robust foundation, revealing key principles and limitations that can channel future efforts. Meticulous incremental enhancements to the methodologies, paired with comprehensive performance evaluations, will be indispensable in unlocking the utilisation of supermask overlaps full potential.

Chapter 5: Conclusion

5.1 Comprehensive Summary and Critical Analysis of Research Findings

This research instituted an exhaustive investigation of supermask overlaps in neural networks, driven by the imperative of fortifying the efficiency of a continual learning framework while addressing the critical challenge of catastrophic forgetting. By investigating the MNIST dataset and its variations, the study conscientiously identified the representational sharing tendencies that neural networks rely on to facilitate generalisation across varied tasks, even in contempt of input data transformations.

There were three key research objectives:

1. The primary objective entailed determining the extent of representational overlap of supermasks through task similarity across distinct tasks in the MNIST dataset variants, despite transformations. This objective was rooted in contemporary research indicating the role of such overlaps in enhancing generalisation and supporting continual learning.

2. The second objective involved proposing and evaluating a novel approach that integrates estimated task similarities through supermask overlaps. The rationale was that explicitly modelling and utilising overlap could accelerate learning efficiency.

3. The tertiary objective focused on iteratively refining the proposed methodology through rigorous comparative analysis against the classic approach across a spectrum of metrics and dataset variants. This was to critically discern the contexts where exploiting overlap confers advantages against conventional techniques.

Moreover, the investigations yielded several crucial findings:

- Experiment 1 and the inceptive procedure of Experiment 2 provided empirical evidence and measurement for extensive supermask overlap across varied MNIST tasks, even after transformations. The consistently high overlaps and statistically indistinguishable neural activations exposed representational sharing facilitating generalisation, aligning with existing literature. However, a critical appraisal revealed sensitivity to input transformations, warranting further principled analysis.

- Experiment 2 offered promising results, with the novel approach requiring consistently less training time compared to the classic methodology across all datasets. This affirmed the hypothesis that integrating task similarities through supermask overlaps can potentially accelerate learning without performance trade-offs.

- Experiment 3, however, provided a vital analytical counterpoint. With additional training iterations modifications and more granular performance metrics, the classic approach exhibited a slight effectiveness edge, underscoring the context-dependent nature of the benefits from modelling overlap.

Critically synthesising the collective insights, it becomes evident that while supermask overlaps are a pivotal asset, their advantages are nuanced, narrow and contingent on the precise techniques involved. The research provided valuable conceptual and empirical contributions, while rigorously delineating limitations that should channel future efforts to unlock the fuller potential of supermask overlaps.

5.2 Critical Appraisal of Key Contributions and Limitations

The granular contributions and inherent limitations stemming from our study are as follows:

1. **Representational Overlap Insights**:

    • **Contribution**: Solid empirical measurement of representational overlap and reuse in neural networks, corroborating and building on existing findings.

    • **Limitation**: Narrow scope of analysed transformations could restrict general applicability.

2. **Novel Learning Approach**:

    • **Contribution**: Significant enhancements in training efficiency, underscoring the potential of exploiting overlaps for learning efficiency.

    • **Limitation**: The study's focus on limited techniques, network architectures and datasets might constrain broader validity.

3. **Methodological Refinements**:

    • **Contribution**: Critical insights into the context-dependent nature of overlap benefits.

    • **Limitation**: The limited scope of metrics explored offers a potentially partial perspective.

4. **MNIST Dataset Dependence**:

    • **Limitation**: Sole reliance on MNIST and its variants risks external validity, despite the dataset's prominence.

5. **Statistical Rigour**:

    • **Limitation**: Predominant use of basic statistical testing, like t-tests, potentially diminishes analytical depth and credibility.

6. **Neural Architectural Constraints**:

    • **Limitation**: Utilising mainly simplistic fully-connected networks might limit the research's reach in terms of representational capacity and wider relevance.

Our research, while making valuable inroads and significant strides, did face analytical constraints highlighting pivotal areas demanding enhancement.

5.3 Future Research Implications and Directions

Our findings and identified gaps illuminate several compelling trajectories for continued exploration:

1. **Dataset Diversification**: Exploring beyond MNIST to encompass real-world datasets will augment external validity and possibly reveal unforeseen challenges.

2. **Architectural Exploration**: Delving into modern, complex neural network architectures could unlock more substantial representational dynamics and insights into supermask overlaps.

3. **Transformational Breadth**: Investigating a wider array of challenging input modifications can further unveil the intricacies of representational overlap.

4. **Metric Expansion**: Incorporating a broader set of evaluation metrics can provide a more holistic understanding of performance nuances.

5. **Statistical Depth**: Employing advanced statistical methodologies could sharpen analytical rigour and provide further credibility to the findings.

6. **Methodological Refinements**: Adaptive and hierarchical integration of task similarities, based on evolving contextual performance, could yield refined and optimised outcomes.

5.4 Final Remark

Conclusively, this research has made valuable inroads in the evolving domain of continual learning, utilising supermask overlaps as a methodological lens. Nonetheless, the exposed limitations signal unexploited potential and channels for systematic enhancements. The pathways illuminated can guide future efforts aiming to fully harness the latent potential of supermask overlaps in efficiently and effectively neutralising catastrophic forgetting and substantially enhancing artificial learning systems' temporal adaptability and intelligence.

Reference List

Agarap , A.F. (2018). Deep Learning using Rectified Linear Units (ReLU). [online] arXiv.org. Available at: https://arxiv.org/abs/1803.08375.

Anselmi, F., Leibo, J., Rosasco, L., Mutch, J., Tacchetti, A. and Poggio, T. (2014). Unsupervised learning of invariant representations with low sample complexity: the magic of sensory cortex or a new framework for machine learning? [online] Available at: https://arxiv.org/pdf/1311.4158.pdf.

Balakrishnan, V., Shi, Z., Law, C.L., Lim, R., Teh, L.L. and Fan, Y. (2021). A deep learning approach in predicting products' sentiment ratings: a comparative analysis. The Journal of Supercomputing. doi: https://doi.org/10.1007/s11227-021-04169-6.

Bank, D., Koenigstein, N. and Giryes, R. (2021). Autoencoders. arXiv:2003.05991 [cs, stat]. [online] Available at: https://arxiv.org/abs/2003.05991.

Ben-Iwhiwhu, E., Nath, S., Pilly, P.K., Kolouri, S. and Soltoggio, A. (2023). Lifelong Reinforcement Learning with Modulating Masks. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.2212.11110.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. arXiv (Cornell University). doi: https://doi.org/10.48550/arxiv.1206.5533.

Bengio, Y., Léonard, N. and Courville, A. (2013). Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1308.3432.

Bengio, Y., Louradour, J., Collobert, R. and Weston, J. (2009). Curriculum learning. Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09. doi: https://doi.org/10.1145/1553374.1553380.

Bishop, C.M. (2006). Pattern Recognition and Machine Learning. Springer.

Blalock, D., Ortiz, J.J.G., Frankle, J. and Guttag, J. (2020). What is the State of Neural Network Pruning? [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.2003.03033.

Bottou, L. (2010). Large-Scale Machine Learning with Stochastic Gradient Descent. Proceedings of COMPSTAT'2010, pp.177–186. doi: https://doi.org/10.1007/978-3-7908-2604-3_16.

Broomhead, D.S. and Lowe, D.J. (1988). Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks. Complex Systems, 2, pp.321–355.

Brynjolfsson, E. and McAfee, A. (2017). The Business of Artificial Intelligence. [online] Harvard Business Review. Available at: https://hbr.org/2017/07/the-business-of-artificial-intelligence.

Bui, T.D., Nguyen, C.V., Swaroop, S. and Turner, R.E. (2018). Partitioned Variational Inference: A unified framework encompassing federated and continual learning. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1811.11206.

Caruana, R. (1997). Multitask learning. Machine Learning, 28(1), pp.41–75. doi: https://doi.org/10.1023/a:1007379606734.

Chaudhry, A., Ranzato, M., Rohrbach, M. and Elhoseiny, M. (2018). Efficient Lifelong Learning with A-GEM. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1812.00420.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P.K., Torr, P.H.S. and Ranzato, M. (2019). On Tiny Episodic Memories in Continual Learning. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1902.10486.

Chen, J. and Ran, X. (2019). Deep Learning With Edge Computing: A Review. Proceedings of the IEEE, [online] 107(8), pp.1655–1674. doi: https://doi.org/10.1109/jproc.2019.2921977.

Chen, Z. and Liu, B. (2018). Lifelong Machine Learning, Second Edition. Synthesis Lectures on Artificial Intelligence and Machine Learning, 12(3), pp.1–207. doi: https://doi.org/10.2200/s00832ed1v01y201802aim037.

Collins, M.D. and Kohli, P. (2014). Memory Bounded Deep Convolutional Networks. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1412.1442.

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B. and Bharath, A.A. (2018). Generative Adversarial Networks: An Overview. IEEE Signal Processing Magazine, 35(1), pp.53–65. doi: https://doi.org/10.1109/msp.2017.2765202.

Dai, B., Zhu, C., Guo, B. and Wipf, D. (2018). Compressing Neural Networks using the Variational Information Bottleneck. [online] proceedings.mlr.press. Available at: https://proceedings.mlr.press/v80/dai18d.html.

Delange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G. and Tuytelaars, T. (2021). A continual learning survey: Defying forgetting in classification tasks. IEEE Transactions on Pattern Analysis and Machine Intelligence, [online] pp.1–1. doi: https://doi.org/10.1109/TPAMI.2021.3057446.

Dettmers, T. and Zettlemoyer, L. (2019). Sparse Networks from Scratch: Faster Training without Losing Performance. arXiv:1907.04840 [cs, stat]. [online] Available at: https://arxiv.org/abs/1907.04840.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. [online] arXiv.org. Available at: https://arxiv.org/abs/1810.04805.

Douillard, A., Chen, Y., Dapogny, A. and Cord, M. (2021). Tackling Catastrophic Forgetting and Background Shift in Continual Semantic Segmentation. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.2106.15287.

Ebrahimi, S., Elhoseiny, M., Darrell, T. and Rohrbach, M. (2020). Uncertainty-guided Continual Learning with Bayesian Neural Networks. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1906.02425.

Eklavya (2019). Kohonen Self-Organizing Maps. [online] Medium. Available at: https://towardsdatascience.com/kohonen-self-organizing-maps-a29040d688da.

Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A.A., Pritzel, A. and Wierstra, D. (2017). PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv:1701.08734 [cs]. [online] Available at: https://arxiv.org/abs/1701.08734.

Finn, C., Abbeel, P. and Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. arXiv:1703.03400 [cs]. [online] Available at: https://arxiv.org/abs/1703.03400.

Frankle, J. and Carbin, M. (2019). The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1803.03635.

French, R. (1999). Catastrophic forgetting in connectionist networks. Trends in Cognitive Sciences, 3(4), pp.128–135. doi: https://doi.org/10.1016/s1364-6613(99)01294-2.

Gale, T., Elsen, E. and Hooker, S. (2019). The State of Sparsity in Deep Neural Networks. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1902.09574.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. [online] Available at: https://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf.

Goodfellow, I., Bengio, Y. and Courville, A. (2016). Deep Learning. MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014). Generative Adversarial Nets. [online] Available at: https://arxiv.org/pdf/1406.2661.pdf.

Goodfellow, I.J., Mirza, M., Xiao, D., Courville, A. and Bengio, Y. (2013). An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. arXiv:1312.6211 [cs, stat]. [online] Available at: https://arxiv.org/abs/1312.6211.

Gray, S., Radford, A. and Kingma, D. (2017). GPU Kernels for Block-Sparse Weights. [online] Available at: https://cdn.openai.com/blocksparse/blocksparsepaper.pdf.

Guo, Y., Yao, A. and Chen, Y. (2016). Dynamic Network Surgery for Efficient DNNs. [online] Neural Information Processing Systems. Available at: https://papers.nips.cc/paper_files/paper/2016/hash/2823f4797102ce1a1aec05359cc16dd9-Abstract.html.

Hadsell, R., Rao, D., Rusu, A.A. and Pascanu, R. (2020). Embracing Change: Continual Learning in Deep Neural Networks. Trends in Cognitive Sciences, [online] 24(12), pp.1028–1040. doi: https://doi.org/10.1016/j.tics.2020.09.004.

Han, S., Pool, J., Tran, J. and Dally, W.J. (2015). Learning both Weights and Connections for Efficient Neural Networks. arXiv:1506.02626 [cs]. [online] Available at: https://arxiv.org/abs/1506.02626.

Hassibi, B. and Stork, D. (1992). Second order derivatives for network pruning: Optimal Brain Surgeon. [online] Neural Information Processing Systems. Available at:

https://proceedings.neurips.cc/paper/1992/hash/303ed4c69846ab36c2904d3ba8573050-Abstract.html.

He, K., Gkioxari, G., Dollar, P. and Girshick, R. (2017). Mask R-CNN. 2017 IEEE International Conference on Computer Vision (ICCV). [online] doi: https://doi.org/10.1109/iccv.2017.322.

He, K., Zhang, X., Ren, S. and Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. 2015 IEEE International Conference on Computer Vision (ICCV). [online] doi: https://doi.org/10.1109/iccv.2015.123.

He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J. and Han, S. (2018). AMC: AutoML for Model Compression and Acceleration on Mobile Devices. doi: https://doi.org/10.48550/arxiv.1802.03494.

Hossain, M.I., Rakib, M., Elahi, M.M.L., Mohammed, N. and Rahman, S. (2022). COLT: Cyclic Overlapping Lottery Tickets for Faster Pruning of Convolutional Neural Networks. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.2212.12770.

Hsu, Y.-C., Liu, Y.-C., Ramasamy, A. and Kira, Z. (2019). Re-evaluating Continual Learning Scenarios: A Categorization and Case for Strong Baselines. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1810.12488.

Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J. and Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and . arXiv:1602.07360 [cs]. [online] Available at: https://arxiv.org/abs/1602.07360.

Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. [online] arXiv.org. Available at: https://arxiv.org/abs/1502.03167.

Janiesch, C., Zschech, P. and Heinrich, K. (2021). Machine learning and deep learning. Electronic Markets, [online] 31. doi: https://doi.org/10.1007/s12525-021-00475-2.

Janowsky, S.A. (1989). Pruning versus clipping in neural networks. Physical Review A, 39(12), pp.6600–6603. doi: https://doi.org/10.1103/physreva.39.6600.

Jordan, M. and Mitchell, T. (2015). Machine learning: Trends, perspectives, and prospects. In: Science.

Kaiser, L., Gomez, A.N., Shazeer, N., Vaswani, A., Parmar, N., Jones, L. and Uszkoreit, J. (2017). One Model To Learn Them All. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1706.05137.

Kemker, R. and Kanan, C. (2018). FearNet: Brain-Inspired Model for Incremental Learning. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1711.10563.

Kim, G., Xiao, C., Konishi, T., Ke, Z. and Liu, B. (2022). A Theoretical Study on Solving Continual Learning. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.2211.02633.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D. and Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences, [online] 114(13), pp.3521–3526. doi: https://doi.org/10.1073/pnas.1611835114.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. Biological Cybernetics, [online] 43(1), pp.59–69. doi: https://doi.org/10.1007/bf00337288.

Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems, [online] 25. Available at: https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

LeCun, Y., Bengio, Y. and Hinton, G. (2015). Deep Learning. Nature, [online] 521(7553), pp.436–444. doi: https://doi.org/10.1038/nature14539.

Lecun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278–2324. doi: https://doi.org/10.1109/5.726791.

LeCun, Y., Denker, J. and Solla, S. (1989). Optimal Brain Damage. [online] Neural Information Processing Systems. Available at: https://proceedings.neurips.cc/paper/1989/hash/6c9882bbac1c7093bd25041881277658-Abstract.html.

Lee, N., Ajanthan, T., Gould, S. and Torr, P.H.S. (2020). A Signal Propagation Perspective for Pruning Neural Networks at Initialization. arXiv:1906.06307 [cs, stat]. [online] Available at: https://arxiv.org/abs/1906.06307.

Li, X., Zhou, Y., Wu, T., Socher, R. and Xiong, C. (2019). Learn to Grow: A Continual Structure Learning Framework for Overcoming Catastrophic Forgetting. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1904.00310.

Lopez-Paz, D. and Ranzato, M. (2017). Gradient Episodic Memory for Continual Learning. arXiv:1706.08840 [cs]. [online] Available at: https://arxiv.org/abs/1706.08840.

Maes, P., Matarić, M.J., Meyer, J.-A., Pollack, J. and Wilson, S.W. (1996). Incremental Self-Improvement for Life-Time Multi-Agent Reinforcement Learning. doi: https://doi.org/10.7551/mitpress/3118.003.0062.

Mallya, A., Davis, D. and Lazebnik, S. (2018). Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1801.06519.

Mancini, M., Ricci, E., Caputo, B. and Bulò, S.R. (2018). Adding New Tasks to a Single Network with Weight Transformations using Binary Masks. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1805.11119.

Marques, J., Falcao, G. and Alexandre, L.A. (2018). Distributed Learning of CNNs on Heterogeneous CPU/GPU Architectures. Applied Artificial Intelligence, 32(9-10), pp.822–844. doi: https://doi.org/10.1080/08839514.2018.1508814.

McClelland, J.L., McNaughton, B.L. and O'Reilly, R.C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. Psychological Review, 102(3), pp.419–457. doi: https://doi.org/10.1037/0033-295x.102.3.419.

McCloskey, M. and Cohen, N.J. (1989). Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. Psychology of Learning and Motivation, pp.109–165. doi: https://doi.org/10.1016/s0079-7421(08)60536-8.

McMahan , B., Moore, E., Ramage, D., Hampson, S. and Arcas, Blaise Agüera y (2016). Communication-Efficient Learning of Deep Networks from Decentralized Data. [online] arXiv.org. Available at: https://arxiv.org/abs/1602.05629.

Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. [online] arXiv.org. Available at: https://arxiv.org/abs/1301.3781.

Mocanu, D.C., Mocanu, E., Stone, P., Nguyen, P.H., Gibescu, M. and Liotta, A. (2018). Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. Nature Communications, 9(1). doi: https://doi.org/10.1038/s41467-018-04316-3.

Molchanov, D., Ashukha, A. and Vetrov, D. (2017). Variational Dropout Sparsifies Deep Neural Networks. [online] proceedings.mlr.press. Available at: https://proceedings.mlr.press/v70/molchanov17a.html.

Molchanov, P., Mallya, A., Tyree, S., Frosio, I. and Kautz, J. (2019). Importance Estimation for Neural Network Pruning. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1906.10771.

Nair, V. and Hinton, G. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. [online] Available at: https://www.semanticscholar.org/paper/Rectified-Linear-Units-Improve-Restricted-Boltzmann-Nair-Hinton/a538b05ebb01a40323997629e171c91aa28b8e2f.

Namhoon, L., Thalaiyasingam, A. and S., T., Philip H. (2018). SNIP: Single-shot Network Pruning based on Connection Sensitivity. arXiv.org. [online] Available at: https://arxiv.org/abs/1810.02340.

Nguyen, C.V., Li, Y., Bui, T.D. and Turner, R.E. (2018). Variational Continual Learning. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1710.10628.

Novikov, A., Podoprikhin, D., Osokin, A. and Vetrov, D. (2015). Tensorizing Neural Networks. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1509.06569.

Pan, S.J. and Yang, Q. (2010). A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering, [online] 22(10), pp.1345–1359. doi: https://doi.org/10.1109/tkde.2009.191.

Parisi, G.I., Kemker, R., Part, J.L., Kanan, C. and Wermter, S. (2019). Continual lifelong learning with neural networks: A review. Neural Networks, [online] 113, pp.54–71. doi: https://doi.org/10.1016/j.neunet.2019.01.012.

Park, S., Lee, J., Mo, S. and Shin, J. (2020). Lookahead: A Far-Sighted Alternative of Magnitude-based Pruning. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.2002.04809.

Pascanu, R., Mikolov, T. and Bengio, Y. (2013). On the difficulty of training Recurrent Neural Networks. arXiv:1211.5063 [cs]. [online] Available at: https://arxiv.org/abs/1211.5063.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L. and Bai, J. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. [online] arXiv.org. Available at: https://arxiv.org/abs/1912.01703.

Radford, A., Narasimhan, K., Salimans, T. and Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training. [online] Available at: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

Ramanujan, V., Wortsman, M., Kembhavi, A., Farhadi, A. and Rastegari, M. (2020). What's Hidden in a Randomly Weighted Neural Network? [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1911.13299. [Software]. Allen Institute for AI. (2020). hidden-networks Available at: https://github.com/allenai/hidden-networks.

Real, R. and Vargas, J.M. (1996). The Probabilistic Basis of Jaccard's Index of Similarity. Systematic Biology, 45(3), pp.380–385. doi: https://doi.org/10.1093/sysbio/45.3.380.

Rebuffi, S.-A., Bilen, H. and Vedaldi, A. (2017). Learning multiple visual domains with residual adapters. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1705.08045.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y. and Tesauro, G. (2019). Learning to Learn without Forgetting by Maximizing Transfer and Minimizing Interference. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1810.11910.

Robins, A. (1995). Catastrophic Forgetting, Rehearsal and Pseudorehearsal. Connection Science, 7(2), pp.123–146. doi: https://doi.org/10.1080/09540099550039318.

Ruder, S. (2017). An Overview of Multi-Task Learning in Deep Neural Networks. [online] arXiv.org. Available at: https://arxiv.org/abs/1706.05098.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986). Learning representations by back-propagating errors. Nature, [online] 323(6088), pp.533–536. doi: https://doi.org/10.1038/323533a0.

Russell, S. and Norvig, P. (2021). ARTIFICIAL INTELLIGENCE : a modern approach, global edition. S.L.: Pearson Education Limited.

Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R. and Hadsell, R. (2016). Progressive Neural Networks. arXiv:1606.04671 [cs]. [online] Available at: https://arxiv.org/abs/1606.04671.

SAZLI, M.H. (2006). A brief review of feed-forward neural networks. Communications, Faculty Of Science, University of Ankara, pp.11–17. doi: https://doi.org/10.1501/0003168.

Schwarz, J., Luketina, J., Czarnecki, W.M., Grabska-Barwinska, A., Teh, Y.W., Pascanu, R. and Hadsell, R. (2018). Progress & Compress: A scalable framework for continual learning. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1805.06370.

Serra, J., Suris, D., Miron, M. and Karatzoglou, A. (2018). Overcoming Catastrophic Forgetting with Hard Attention to the Task. [online] proceedings.mlr.press. Available at: https://proceedings.mlr.press/v80/serra18a.html.

Settles, B. (2009). Active Learning Literature Survey. [online] Available at: https://research.cs.wisc.edu/techreports/2009/TR1648.pdf.

Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. Physica D: Nonlinear Phenomena, 404, p.132306. doi: https://doi.org/10.1016/j.physd.2019.132306.

Shin, H., Lee, J.K., Kim, J. and Kim, J. (2017). Continual Learning with Deep Generative Replay. arXiv:1705.08690 [cs]. [online] Available at: https://arxiv.org/abs/1705.08690.

Strom, N. (1997). Sparse connection and pruning in large dynamic artificial neural networks. doi: https://doi.org/10.21437/eurospeech.1997-708.

Swaroop, S. (2022). Probabilistic Continual Learning using Neural Networks. [online] Available at: https://api.repository.cam.ac.uk/server/api/core/bitstreams/cf6e4ecc-b9ed-4f34-8c29-d76f42bf1466/content.

Tanaka, H., Kunin, D., Yamins, D. and Ganguli, S. (2020). Pruning neural networks without any data by iteratively conserving synaptic flow. [online] Available at: https://arxiv.org/pdf/2006.05467.pdf.

Theis, L., Korshunova, I., Tejani, A. and Huszár, F. (2018). Faster gaze prediction with dense networks and Fisher pruning. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1801.05787.

Thrun, S. (1998). Lifelong Learning Algorithms. Learning to Learn, pp.181–209. doi: https://doi.org/10.1007/978-1-4615-5529-2_8.

Van de Ven, G.M. and Tolias, A.S. (2019). Three Scenarios for Continual Learning. arXiv:1904.07734 [cs, stat]. [online] Available at: https://arxiv.org/abs/1904.07734.

Wang, Y., Yao, Q., Kwok, J. and Ni, L.M. (2019). Generalizing from a Few Examples: A Survey on Few-Shot Learning. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1904.05046.

Wortsman, M., Ramanujan, V., Liu, R., Kembhavi, A., Rastegari, M., Yosinski, J. and Farhadi, A. (2020). Supermasks in Superposition. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.2006.14769. [Software] RAIVN Lab. (2021). SupSup Available at: https://github.com/RAIVNLab/supsup.

Yadav, P. and Bansal, M. (2023). Exclusive Supermask Subnetwork Training for Continual Learning. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.2210.10209.

Yao, Y. and Doretto, G. (2010). Boosting for transfer learning with multiple sources. [online] IEEE Xplore. doi: https://doi.org/10.1109/CVPR.2010.5539857.

Yoon, J., Yang, E., Lee, J. and Hwang, S.J. (2018). Lifelong Learning with Dynamically Expandable Networks. arXiv:1708.01547 [cs]. [online] Available at: https://arxiv.org/abs/1708.01547.

Zang, C., Friswell, M.I. and Imregun, M. (2003). Structural Health Monitoring and Damage Assessment Using Measured FRFs from Multiple Sensors, Part II: Decision Making with RBF Networks. Key Engineering Materials, 245-246, pp.141–148. doi: https://doi.org/10.4028/www.scientific.net/kem.245-246.141.

Zenke, F., Poole, B. and Ganguli, S. (2017). Continual Learning Through Synaptic Intelligence. arXiv:1703.04200 [cs, q-bio, stat]. [online] Available at: https://arxiv.org/abs/1703.04200.

Zhang, X., Zou, J., He, K. and Sun, J. (2016). Accelerating Very Deep Convolutional Networks for Classification and Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(10), pp.1943–1955. doi: https://doi.org/10.1109/tpami.2015.2502579.

Zhang, Y. and Yang, Q. (2017). A Survey on Multi-Task Learning. arXiv (Cornell University). doi: https://doi.org/10.48550/arxiv.1707.08114.

Zhou, H., Lan, J., Liu, R. and Yosinski, J. (2019). Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask. doi: https://doi.org/10.48550/arxiv.1905.01067.