Methods for Deep Transfer Learning and Knowledge Transfer in the NeuCube Brain-Inspired Spiking Neural Network

Yongyao Tan

A thesis submitted to

Auckland University of Technology

in partial fulfillment of the requirements for the degree of

Master of Computer and Information Sciences (MCIS)

2020

Faculty of Design & Creative Technologies

School of Computing and Information Sciences

Supervisors:

Professor Nikola Kasabov

Doctor Maryam Doborjeh

Abstract:

With the increasing number of computational systems based on continuous streams of information, progressively learning and accommodating new knowledge in a more efficient manner becomes a long-standing challenge. This thesis proposes methods employing a Brain-Inspired Spiking Neural Network (BI-SNN) architecture for transfer learning scenarios. The proposed transfer learning approaches were experimentally validated using a benchmark brain data related to upper limb movement. The results showed that the proposed methods have the capability to effectively learn new knowledge by retaining and reusing previously learned knowledge, resulting in a better accuracy of classification (up to 88.89%) when compared with non-transfer learning methods. Further, a new deep knowledge representation approach is proposed and developed, which allows extracting spatial temporal rules from deep knowledge, enabling a better interpretation of learning patterns in the SNN models and evolution trace of knowledge during transfer learning.

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
ATL	Active Transfer Learning
AwAR	Active Weighted Adaptation Regularization
AER	Address-Event Representation
BI-SNN	Brain-Inspired Spiking Neural Network
BCI	Brain-Computer Interfaces
BSA	Ben Spike Algorithm
CFNs	Controlled Forgetting Networks
CWR	Copy Weight with Reinit
deSNN	Dynamic evolving SNN classifier
EWC	Elastic Weight Consolidation
EEG	Electroencephalogram
fMRI	Functional Magnetic Resonance Imaging
FWET	Feature Weighted Episodic Training
GDM	Growing Dual-Memory
G-EM	Growing Episodic Memory
G-SM	Growing Semantic Memory
IID	Independent and Identically Distributed
LIFM	Leaky Integrate and Fire Model

LDP	Long-term Potentiation
LwF	Learning without Forgetting
LTD	Long-term Depression
MNI	Montreal Neurological Institute
RO	Rank-order
SI	Synaptic Intelligence
SPAN	Spike Pattern Association Neuron
SITAL	Selective Instance Transfer with Active Learning
SIITAL	Selective Informative Instance Transfer with Active Learning
SNNs	Spiking Neural Networks
SVN	Support Vector Machine
STDP	Spike Timing Dependent Plasticity
STR	Spatial Temporal Rules
TBR	Threshold Based Encoding
wAR	Weighted Adaptation Regularization

Attestation of Authorship

“I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgments), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning”.

Signature

Date 23/01/2021

Acknowledgments

I have the pleasure to acknowledge the support of many people who have supported, inspired and helped me over this year. First and foremost, I am very grateful to my primary supervisor Professor Nikola Kasabov for his support and inspiration. I would also like to express my appreciation to my secondary supervisor Doctor Maryam Doborjeh for her guidance and support. I am also very thankful to Kaushalya Kumarasinghe, the member of Knowledge Engineering and Discovery Research Institute, who helped me at the beginning of my study.

Chapter 1 Introduction

1.1 Rationale and Motivation

Large amounts of data are produced in the form of continuous streams and the distribution behind these produced data may be changed over time. Consequently, adaptive and scalable algorithms are required to capture such changes and leverage prior knowledge in order to improve the learning performance of target domains and avoid becoming obsolete. The ability of continually learning over long time spans and transferring knowledge across domains, referred to as transfer learning, is crucial for the development of real-life applications when processing continuous streams of information.

The majority of current transfer learning research has been carried out in conventional machine learning techniques, such as, support vector machine (SVM) (Bruzzone, & Marconcini, 2009), and deep neural networks (Rusu et al., 2016; Li, Parikh, & He, 2018). However, these methods often model spatial and temporal components separately without considering informative spatio-temporal correlations in the data (Kasabov, 2019).

Compared to conventional machine learning methods, Spiking Neural Networks (SNNs) are promising computational paradigms for modelling complex information such as Spatio-temporal Brain Data (STBD) (Kasabov, 2019). However, few studies have thoroughly investigated transfer learning in SNNs in a systematic way, even

though they are naturally adaptive to changing environments. The challenge now is to develop new SNN algorithms and methods for the efficient learning of STBD in a transfer learning manner.

This thesis covers this research gap by adapting Brain-Inspired Spiking Neural Network (BI-SNN) architecture in a transfer learning scenario, with the goals of learning new information from different tasks or subjects sequentially, while retaining the knowledge from previously learned tasks or subjects without forgetting. This research also contributes to an improved level of interpretability of learning patterns in SNN models through deep knowledge representation. The task to task and subject to subject transfer learning case studies here use real-life Electroencephalogram (EEG) dataset which was measured prior to this study by New Zealand College of Chiropractic and Aalborg University under the ethical approval of the local ethics committee (N-20130081) who are acknowledged in this thesis.

1.2 Aims of this Thesis and Research Questions

The primary aims of this thesis are summarised as follows:

1) Proposal of novel deep transfer learning methods in BI-SNN.
    - To develop an incremental learning method based on SNN models
    - To develop a transfer learning method based on SNN models
    - To develop connection weights pruning techniques for transfer learning in SNN models

- To develop neuron aggregation technique for transfer learning SNN models

2) Empirical study of the proposed transfer learning approaches on task to task transfer learning in one subject.
    - To design optimal transfer learning in SNN models that can learn sequentially presented tasks
    - To achieve an improved classification accuracy on previous learned tasks while learning new tasks
    - To interpret the learning patterns in SNN models captured during the learning process
3) Empirical study of the proposed transfer learning approaches on subject to subject transfer learning.
    - To design optimal transfer learning in SNN models that can learn sequentially presented subjects
    - To achieve an improved classification accuracy on previously learned subjects while learning new subjects
    - To interpret the learning patterns in SNN model captured during the learning process
4) Development of a new method for knowledge representation in BI-SNN and for tracing the evolution of knowledge during transfer learning.
    - To develop a new method for dynamic spatio-temporal rule extraction of patterns generated during supervised learning in SNN models

- To improve the level of interpretability of learning patterns in the SNN models
- To develop a method for tracing the evolution of knowledge during transfer learning

During the progression of this thesis, the following research questions have been addressed:

Q1. How to develop a transfer learning technique for a Spiking Neural Network framework?

Q2. How to deal with non-stationarity among different domains so that better transfer learning performance can be obtained across domains?

Q4. How to improve the level of interpretability and understanding of learning patterns in a successful transfer learning in SNN models?

1.3 Thesis Structure

This thesis consists of eight chapters which are outlined as follows:

Chapter 1 states the research rationale, motivations, and research questions.

Chapter 2 reviews the research about the definitions and existing approaches for incremental learning and transfer learning and the relationship between these two

learning techniques. This section is then followed by a review on deep knowledge representation in order to gain further understanding of the learning process.

Chapter 3 discusses the techniques of information encoding, computational model of a spiking neuron and learning in SNN models. Then, this chapter introduces a BI-SNN architecture, namely NeuCube for modelling spatio-temporal data.

Chapter 4 reveals the methodology for the study and is proposing two new training approaches including “incremental training” and “transfer learning”, called ImSNN and TrSNN respectively, along with three additional algorithms that can be combined with the proposed transfer learning approaches, including two connection weights pruning algorithms, and one neuron aggregation algorithm.

Chapter 5 proposes a transfer learning approach for the scenario of task to task transfer learning in one subject. It represents the dataset that was used in this research.

Chapter 6 proposes transfer learning algorithms across multiple subjects utilising the SNN architecture under investigation.

Chapter 7 proposes a new algorithm for knowledge representation based on the NeuCube SNN architecture and demonstrates it for knowledge discovery through extracting and analyzing spatial temporal rules (STR) for task to task and subject to subject transfer learning models as introduced in the previous chapters.

Chapter 8 summarises the key findings, contributions, limitations and future works for this research.

Figure 1-1 illustrates a bird's-eye view of the thesis and its different components towards addressing the research questions.

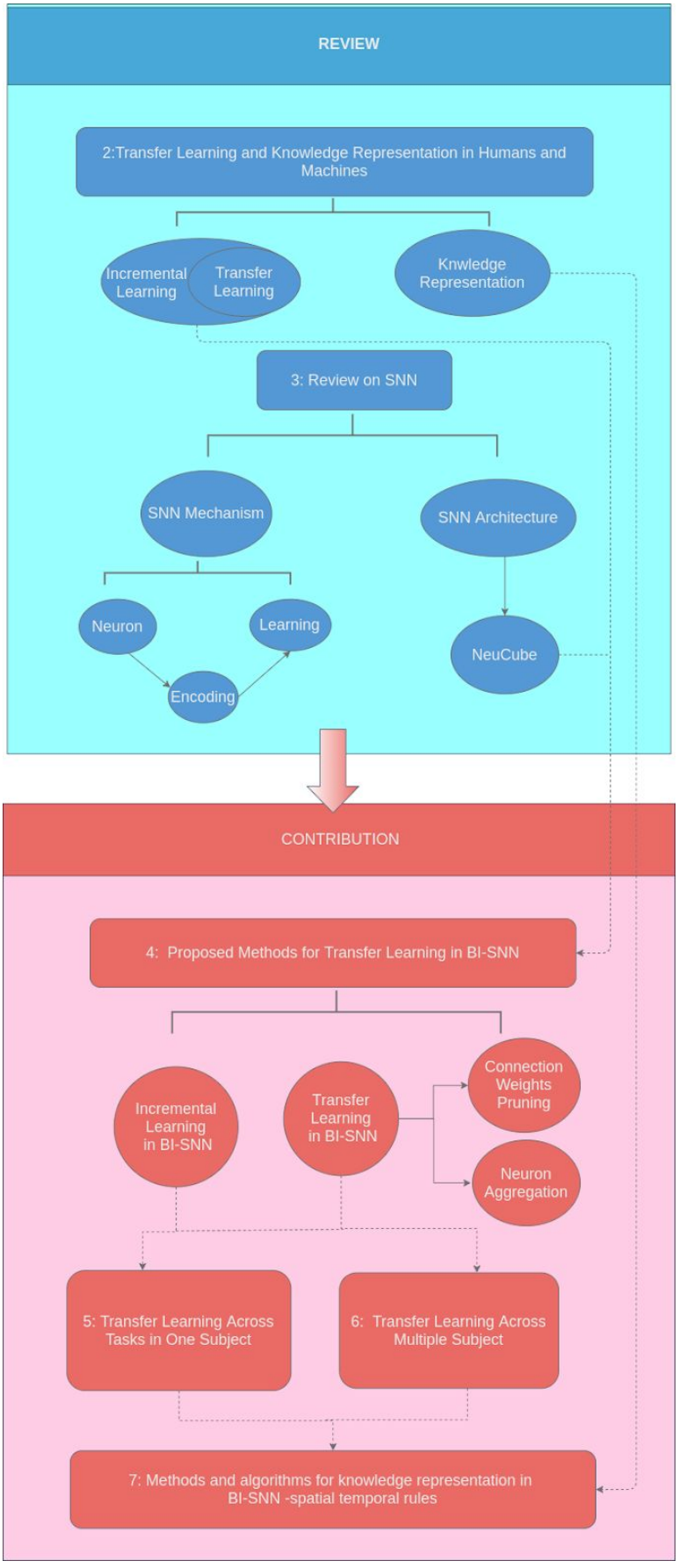

Figure 1-1 A bird's-eye view of the thesis structure.

Chapter 2 Incremental and Transfer Learning and Knowledge Representation in Humans and Machines

2.1 Introduction

This chapter will first review computational approaches of learning which include incremental learning for overcoming catastrophic forgetting problems in Section 2.2 and transfer learning for the reuse of knowledge during learning of new tasks for one subject or the same task across different subjects in Section 2.3. Then, in Section 2.4, deep knowledge representation will be discussed as the main technique for improving the level of interpretability of learning patterns in the SNN models.

2.2 Incremental Learning

Humans and animals exhibit an astonishing ability to learn in a lifelong fashion by incrementally acquiring, accommodating, refining new knowledge and skills, and transferring them across domains (Cichon, & Gan, 2015; Bremner, Lewkowicz & Spence, 2012). The ability to continuously adapt to new knowledge over time without forgetting crucial information from prior learned experiences can be referred to as incremental learning (Parisi, Kemker, Part, & Wermte, 2019). Thus, the capacity of incremental learning is essential for the development of computational systems performing in the real world. However, the tendency for previously learned information being interfered by newly learned knowledge remains a key challenge for

computational models regarding incremental learning. This phenomenon is called catastrophic forgetting or interference (McClelland, McNaughton, & O'Reilly, 1995; McCloskey & Cohen, 1989).

2.2.1 Incremental Learning Approaches

Numerous approaches for incremental learning that mitigate catastrophic forgetting have been explored in the literature. They can be categorised into three groups, regularization approaches, dynamic architectures, and complementary learning systems and memory replay (Parisi, Kemker, Part, & Wermte, 2019). Figure 2-1 presents a venn diagram of some popular incremental learning strategies leveraging ideas from these three categories. These strategies are explained in the following subsections.

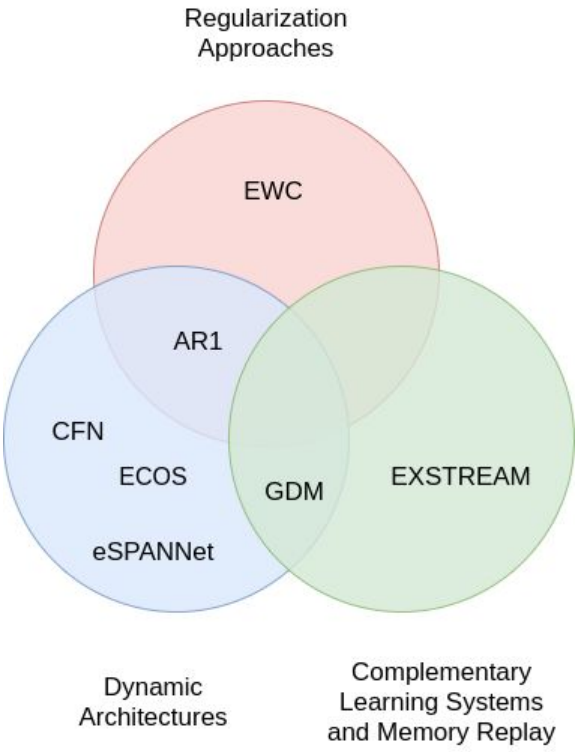

Figure 2-1 Venn diagram of some of the incremental learning strategies: EWC (Kirkpatricka, et al., 2017), AR1 (Maltoni, & Lomonaco, 2019), CFN (Allred, & Roy 2020), ECOS (Kasabov, 2007),

eSPANNet (Kumarasinghe, Taylor, & Kasabov, 2019), GDM (Maltoni, & Lomonaco, 2019), and EXSTREAM (Hayes, Cahill, & Kanan, 2019).

A. Regularization Approaches:

Regularization approaches alleviate catastrophic forgetting by selectively constraining the neural weights, which are vital to retain connections for past memories (Parisi, Kemker, Part, and Wermte, 2019).

Elastic weight consolidation (EWC) (Kirkpatricka, et al., 2017) is a popular regularization strategy attempting to alleviate catastrophic forgetting in two contexts, supervised and reinforcement learning. In this approach, a quadratic penalty was applied to constrain the amount of change of important parameters which are important for old tasks. Given a two tasks scenario, task A and task B, to reduce the magnitude of weight θ changes when training in a new task B, a modified cost function with a regularization term is given by:

$$L(\theta) = L_B(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2 \qquad (2\text{-}1)$$

Where L_B is the loss for task B, λ is the regularization strength, and F is the diagonal of the fisher information matrix.

At the intersection between architectural and regularization strategies, the AR1 model was proposed for single-incremental-task scenarios (Maltoni, & Lomonaco, 2019). Their approach is composed of two components, a modified copy weight with reinit (CWR) (Lomonaco, & Maltoni, 2017), denoted as CWR+, and synaptic intelligence (SI) regularization constraint (Zenke, Poole, & Ganguli, 2017). The

experiment results on CORe50 (Lomonaco, & Maltoni, 2017) and CIFAR-100 (Krizhevsky, 2009) benchmark datasets show that AR1 model outperformed existing regularization approaches such as learning without forgetting (LwF) (Li, & Hoiem, 2016), EWC and SI.

B. Dynamic Architectures

This method expands neural architecture dynamically to accommodate novel neural resources in response to new knowledge, for example, by retraining with an increased number of neurons or network layers (Parisi, Kemker, Part, and Wermte, 2019).

Controlled Forgetting Networks (CFNs) (Allred, & Roy 2020) is one of the architectural strategies proposed and inspired by biological dopamine signals. The modified version of STDP learning rule, called heterogeneously modulated STDP learning, was introduced to perform isolated adaptation in the synapses of neurons related to the novel information while retaining the knowledge from previous tasks. This approach was experimentally validated using the MNIST dataset. The results show that CFN allows the training of spiking neural network models with less forgetting.

In (Kasabov, 2007), the principle of evolving connectionist systems (ECOS) was introduced and later developed and applied for many applications. An ECOS evolves and adapts its structure and functionality from incoming data by generating new neurons to capture new patterns from the incoming data or by adapting existing

neuronal connections to accommodate new data. ECOS allows for fuzzy rule extraction from the incrementally evolved structures. A continuation of this work was the proposed dynamic evolving SNN (deSNN) as discussed in the next chapter (Kasabov et al, 2013).

Kumarasinghe, Taylor and Kasabov (2019) proposed the combination of Spike Pattern Association Neuron (SPAN) model with a computational interpretation of a 'population vector' in order to address the non-stationarity and high trial-to-trial variability of current Brain Computer Interfaces (BCI) applications. This approach, referred to as eSPANNet model, enables incremental learning from incoming training data and online prediction for single-trial BCI. Reported results on the finger flexion prediction dataset from the fourth BCI competition show that eSPANNet allows a higher classification performance compared to several classification approaches and a better approximation of the actual movement signal compared to several other regression analysis methods.

C. Complementary Learning Systems and Memory Replay

Complementary learning systems aim to model memory consolidation and retrieval in which past information is periodically replayed to the model for protecting consolidated knowledge while alleviating the memorization and generalization for complementary tasks (Parisi, Kemker, Part, and Wermte, 2019).

Leveraging ideas from architectural and memory replay strategies, Maltoni and Lomonaco (2019) proposed a Growing Dual-Memory (GDM) architecture for learning

spatiotemporal representations from videos for lifelong learning scenarios. This approach consists of two growing recurrent self-organizing networks that dynamically adapt the number of neurons and synapses, growing episodic memory (G-EM) and growing semantic memory (G-SM). Reported experiments show that the proposed method significantly outperforms current lifelong learning methods in three different incremental learning scenarios with the CORe50 benchmark dataset.

Hayes, Cahill, and Kanan (2019) proposed an ExStream algorithm for memory-efficient rehearsal. In contrast to the full rehearsal approach that eliminates catastrophic forgetting by learning a mixture of all prior samples with new samples, ExStream algorithm stores a smaller number of prototypes that capture most of the intra-class variance instead. Experiments reported good results on four different paradigms, requiring less memory usage and computation compared with full rehearsal approach and other streaming clustering methods.

While numerous algorithms have been developed to address incremental learning tasks, it differs significantly from a richer set of learning capabilities in humans and animals. Learning in a continual manner goes beyond the ability to accommodate new knowledge incrementally, importantly, benefiting from generalized knowledge and skills across domains and tasks (Parisi, Kemker, Part, and Wermte, 2019). Figure 2-2 presents a schematic view of the main difference between traditional machine learning, incremental learning and transfer learning. The next section will discuss transferring the learned knowledge and skills across multiple domains.

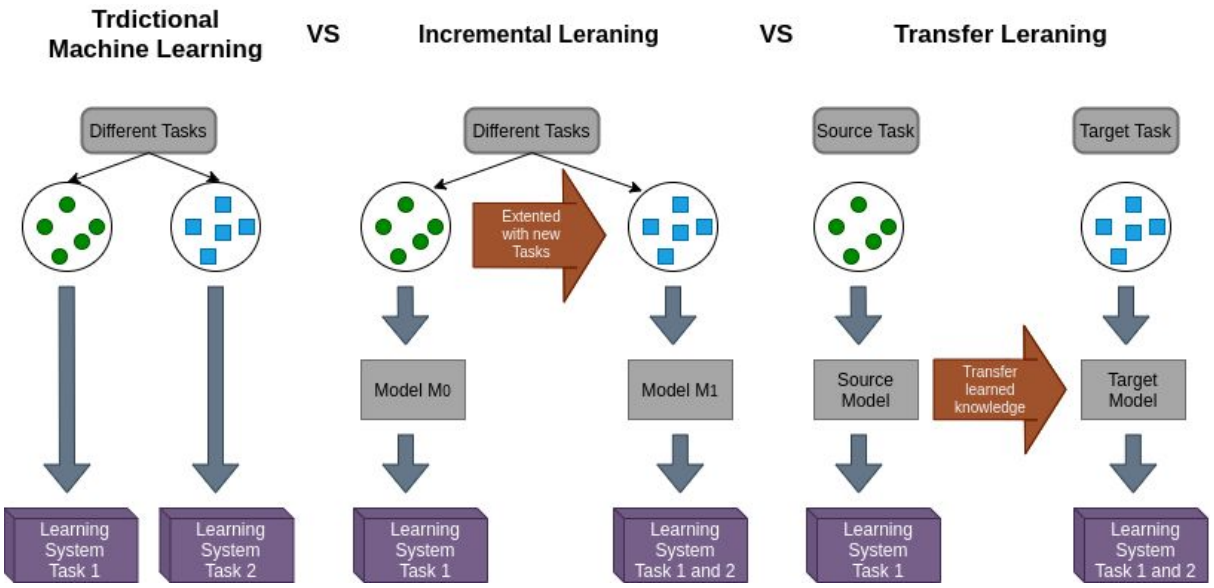

Figure 2-2 Schematic view of the main difference between traditional machine learning, incremental learning (Section 2.2), and transfer learning (Section 2.3).

2.3 Transfer Learning

Transfer learning which aims to leverage the previously acquired knowledge from source domains and then apply this knowledge to a new target domain, can be considered as a beneficial solution to reduce the expensive and time-consuming data collection efforts. The definition of transfer learning is given following the notations introduced by Pan and Yang (2009). A domain D is composed of two terms: feature space χ and its marginal probability distribution $P(X)$, which is denoted by $D = \{\chi, P(X)\}$. Subsequently, given a specific domain D, its task T is defined by a label space Y and an predictive function $f(.)$, which can be to learn the feature and label pairs $\{x_i, y_i\}$, where $x_i \in \chi$ and $y_i \in \mathrm{Y}$. From the perspective of probability, the predicted function $f(.)$ can be written as $P(Y|X)$.

Based on the notations defined above, the definition of transfer learning can be defined as follows:

> Definition 1. "Given a source domain D_S and learning task T_S, a target domain D_T and and learning task T_T, transfer learning aims to help improve the learning of the target predictive function $f_T(.)$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$" (Pan, & Yang, 2009).

The above definition, the target and source domains are not the same ($D_S \neq D_T$) implies that either the feature spaces ($\chi_S \neq \chi_T$) are different, the marginal probability distributions ($P_S(X) \neq P_T(X)$) are different, or both. Similarly, given specific domains D_S and D_T, two different learning tasks ($T_S \neq T_T$) refers to either mismatch label spaces between domains ($\mathrm{Y}_S \neq \mathrm{Y}_T$), mismatch conditional probability distributions between domains ($P(X_S|Y_S) \neq P(X_T|Y_T)$), or both. In the context of traditional machine learning, the target and source domains are the same ($D_S = D_T$), and their learning tasks are the same ($T_S = T_T$). The common notations used in this study is
summarized in Table 2-1.

Table 2-1 Summary of commonly used notation.

Notation	Description	Notation	Description
Subscript S	Denotes source	χ	Feature space
Subscript T	Denotes target	$P(X)$	Marginal distribution
D	Domain data	Y	Label space
T	Learning task	$P(Y\|X)$	Conditional distribution

2.2.2 Transfer Learning Categories

Based on whether the source and target domains and tasks are identical or not, transfer learning can be broadly divided into three main categories, namely inductive transfer learning, transductive transfer learning, and unsupervised transfer learning.

A. Inductive Transfer Learning

Inductive transfer learning refers to the scenario where knowledge is transferred across different but related source and target tasks regardless of whether the source and target domains are identical or not. Moreover, the label information of the target domain is available in inductive transfer learning (Pan, & Yang, 2009). Based on the availability of label information from the source domain, inductive transfer learning can be further categorized into two categories:

- When a large amount of labeled data is available from the source domain, inductive transfer learning is similar to Multi-task learning (Alamgir, Grosse-Wentrup, & Altun, 2010).
- When the label information is unknown for the source domain, inductive transfer learning is similar to Self-taught learning (Raina, Battle, & Lee H, 2007)

B. Transductive Transfer Learning

In the setting of transductive transfer learning, the source and target are represented in different domains, but the same task. In this case, a large amount of

source domain labelled trials is available, while the label information is unknown for target domains (Pan, & Yang, 2009). According to the consistency between the source and the target feature spaces and the marginal probability distributions of the input data, transductive transfer learning can be further categorized into two categories:

- When there are non-identical label spaces between domains, $X_S \neq X_T$, this scenario is referred to as heterogeneous transfer learning (Weiss, Khoshgoftaar, & Wang, 2016).
- When the source and target domains are represented in different feature spaces, $X_S = X_T$, but different marginal distributions, $P_S(X) \neq P_T(X)$, this scenario is termed as homogeneous transfer learning (Weiss, Khoshgoftaar, & Wang, 2016).

C. Unsupervised Transfer Learning

Unsupervised transfer learning aims to transfer knowledge across different but related source and target tasks. In this transfer learning method, the label information of both target source and domain instances are not available (Pan, & Yang, 2009).

2.2.3 Transfer Learning Approaches

Based on which types of knowledge that can be transferred across domains or tasks, approaches to transfer learning can be categorised into four different groups. These approaches are explained in the following subsections.

A. Instance-based Transfer Learning

This instance-representation-based approach is based on the assumption that some similar features are shared between source and target domains and the performance of the target prediction function can be improved by reusing certain parts of the source domain labeled data for learning the target domain (Azab, Toth, Mihaylova, & Arvaneh, 2018). The two major techniques in this context are instance re-weighting (Jiang, & Zhai, 2007) and importance sampling.

An example of instance-based transfer learning, called improved active transfer learning (ATL), was proposed by Hossain, Khosravi and Nahavandhi (2016) to reduce non-stationarities between subjects. The main principle of active transfer learning is that only the most informative samples are labeled such that the learning performance of a new subject or a new task is expected to be improved. They also proposed two extended algorithms of ATL. The first one was called Selective Instance Transfer with Active Learning (SITAL), which aims to select samples from source domains that have a similar distribution with the target domain. The second algorithm was classed Selective Informative Instance Transfer with Active Learning (SIITAL), which selects samples from source domain that have both high classification accuracy and normalized entropy. The proposed algorithm is promising when there is a lower quantity of data from the target subject, as it was suggested to reduce the calibration effort.

Another approach based on the principle of weighted adaptation regularization (wAR) and active learning, namely active weighted adaptation regularization (AwAR), was proposed by Wu, Lawhern, Hairston, and Lance (2016) to reduce non-stationarity between headsets in BCI. wAR is applied under the assumption when there is a large amount of labeled data from the previous headset and this data can be used to improve the learning performance for a different headset. They also proposed to integrate wAR with active learning, which selects the most beneficial target samples for labelling. This integrated algorithm can achieve a desired classification accuracy, given a small number of labeled samples required from the new headset, thus making AwAR more suitable for wide-scale applications.

B. Feature-representation Transfer Learning

This feature-representation-based approach focuses on transferring the knowledge via the adjustment and the transformation of feature representation, such that the knowledge can be transferred across domains. Specifically, by constructing a new feature representation for the target domain, the performance of the target tasks can be improved (Azab, Toth, Mihaylova, & Arvaneh, 2018). According to different methods of mapping the original feature to the new feature representation, feature-based approaches can be further divided into asymmetric and symmetric feature-based transfer learning (Kulis, Saenko, & Darrell, 2011; Pan, Tsang, Kwok, & Yang, 2009). Asymmetric approaches aim to map the features from the source domain to the target domain, while symmetric approaches try to create a common latent feature space for both source and target domains.

He and Wu (2019) proposed a novel Electroencephalogram (EEG) trial alignment approach in the Euclidean space across different subjects such that the similarity of data distributions between different subjects can be increased. This Euclidean-space alignment approach only required unlabelled EEG data from the new subject, and it can be used as a preprocessing step before any signal processing, feature extraction, and machine learning algorithms. The experiment results showed that the proposed approach led to enhanced computation when comparing with the Riemannian space covariance matrix alignment approach.

Recently, the proposed different set domain adaptation approach (2020) was presented to be an extension of the Euclidean-space alignment approach, which considers different domain adoption scenarios for both task-task and subject-to-subject transfer, for example source and target subjects have different label spaces and feature spaces. They introduced a label alignment approach to align the label space across different source and target domains. A higher performance can also be achieved when integrating with other domain adoption approaches.

C. Classifier-based Transfer Learning

The previous two approaches of transfer learning aim to transfer the knowledge in the data level, while the classifier-based approach mainly focuses on transferring the knowledge in the parameter level. An assumption behind the classifier-based transfer learning is that there are some shared parameters or prior distributions between the prediction function of the source and target tasks. With the shared

parameters or priors, the performance of the target prediction function is expected to be improved (Azab, Toth, Mihaylova, & Arvaneh, 2018).

For instance, Jayaram, Alamgir, Altun, Scholkopf, and Grosse-Wentrup (2016) proposed a general framework for transfer learning in the context of BCIs that is applicable to any arbitrary feature space, as well as a regression estimation method. Reported results on both motor imagery and a novel cognitive paradigm showed that the proposed framework outperformed other comparable methods with both session-to-session and subject-to-subject scenarios.

Cui, Xu and Wu (2019) proposed a feature weighted episodic training (FWET) approach, which does not require any calibration data from the new subject, eliminating the calibration requirement completely. FWET consists of two parts: feature weighting and episodic training. Feature weighting is used to learn the weights for each feature automatically according to the importance of different features, while episodic training is used for domain generalization. Experiments on driver drowsiness estimation showed that FWET can achieve better generalization performance, given no calibration data from the new subject, thus making FWET more suitable for plug-and-play BCI applications.

D. Relational-based Transfer Learning

The relational-based transfer learning approach mainly focuses on the problem that data in the source and target domains are not independent and identically distributed (IID), but have some similar relational patterns. Thus, by extracting some common

relationships among the data, knowledge can be transferred across tasks (Azab, Toth, Mihaylova, & Arvaneh, 2018). Statistical relational learning techniques are well-known techniques using this approach (Mihalkova, Huynh,& Mooney, 2007; Davis, & Domingos, 2008).

It can be concluded from the literature that the majority of current transfer learning research has been carried out in traditional machine learning techniques. However, these methods often model spatial and temporal components separately without considering informative spatio-temporal correlations in the data (Kumarasinghe, Taylor, & Kasabov, 2019). Compared to conventional machine learning methods, Spiking Neural Networks (SNNs) are promising computational paradigms for modelling complex information such as spatio-temporal data (Kasabov, 2019), which will be discussed in Chapter 3.

2.4 Deep Knowledge Representation

Deep learning in the brain is achieved through processing information, either triggered by external stimuli, or by inner processes, for example, visual auditory, tactile, gustatory and olfactory, complex neural network connections between neurons in space and time are formed across the whole brain. The patterns that are formed by these connections represent deep knowledge (Kasabov, 2019).

Deep knowledge is defined by Kasabov (2019) to refer to a sequence of events that happen in different spatially located parts of the brain, activated at different times,

constituting dynamically changing and Informative knowledge deep in time and space through symbolic and/or numerical expressions.

More specifically, given a set of events that form a neural trajectory $E = \{E_1, E_2, ..., E_n\}$, each E_i can be represented as the following format:

$$E_i = (F_i, S_i, T_i)$$

Where F_i is a function that trigger event changes; S_i is the location of the activity at time T_i.

Deep knowledge can also be represented in several forms. One way to represent deep knowledge through a deep crisp rule. Given the activation level of three neural clusters $(S = \{S_1, S_2, S_3\})$ over three-time bins $(T = \{T_1, T_2, T_3\})$, the crisp rule can be defined as follow:

IF (event E_1 : function F_1, location around S_1, time about T_1)

AND (event E_2 : function F_2, location around S_2, time about T_2)

AND (event E_3 : function F_3, location around S_3, time about T_3)

THEN (The pattern event/task/process Q is recognised)

One limitation of crisp rules is that it is only suitable for the case when the activation of exact clusters of neurons happens at exact times in their sequence. Another form of spatial-temporal rule is the deep fuzzy rule (Zadeh, 1965), which allows for the pattern Q to be recognised even If the slightly different cluster neurons are activated at slightly different times. An example of deep fuzzy rule with corresponding fuzzy values W, which represented by their membership functions, is given as follow:

IF (event E_1: function F_1, location around S_1, time about T_1, probability about P_1, strength is W_1)

AND (event E_2: function F_2, location around S_2, time about T_2, probability about P_2, strength is W_2)

AND (event E_3: function F_3, location around S_3, time about T_3, probability about P_3, strength is W_3)

THEN (The pattern event/task/process Q is recognised)

2.4.1 Knowledge Granularity

When considering a sequence of events that happen in a similar location at a similar time, a suitable level of spatial and temporal scale is required for the rule extraction. Different spatial and temporal scales represent different knowledge information, forming a granularity level of deep representation (Kasabov, 2019).

The terms of time resolution or time depth and spatial resolution or spatial depth define spatial and temporal scales to represent the knowledge. The temporal depth refers to the size of time bin T_i of the brain signals, ranging from milliseconds to second (Kasabov, 2019). The spatial depth of knowledge is defined by the spatial cluster of neurons associated with different anatomical levels (Kasabov, 2019). For example, at a higher level of the hierarchy, neuronal clusters spatially located in two hemispheres can be considered as two distinct granules. In a deeper hierarchy level, neuronal clusters spatially located in different lobes of the brain can be considered as a separate knowledge granule. At the next level, the knowledge can further

scale-up to the different cellular areas in each lobe. A proper level of granularity for a given task is essential and hard to be defined and it depends on different tasks and problems, even measured data (Kumarasinghe, Kasabov, & Taylor, 2020).

2.5 Chapter Summary

This chapter reviewed two computational approaches of learning (incremental learning and transfer learning), which remain a long-standing challenge for computational models. Then, a review on deep knowledge representation was presented. In the next chapter, Brain-Inspired Spiking Neural Network principles and models are discussed.

Chapter 3 Spiking Neural Networks

3.1 Introduction

The Brain-Inspired Spiking Neural Networks (SNNs) are a promising computational paradigm that consist of artificial neurons with interconnected structure, where internal information is represented as trains of spikes and learned in an adaptive and self-organising manner, similar to how a biological neuron functions (Izhikevich, 2006; Brette, et al., 2007). This inherent nature of the spiking neuron has the capacity to model complex information such as spatio-temporal data (Kasabov, 2019).

So far, numerous applications of SNNs have been developed, such as EEG data modelling (Kasabov, & Capecci, 2015; Doborjeh, Kasabov, Doborjeh, & Sumich, 2018), Functional Magnetic Resonance Imaging (fMRI) data modelling (Kasabov, Doborjeh, & Doborjeh, 2017), Brain-Computer Interfaces (BCI) (Taylor et al., 2014; Hu, Hou, Chen, Kasabov, & Scott, 2014; Kumarasinghe, Owen, Taylor, Kasabov, & Kit, 2018), multimodal audio-visual information processing (Wysoski, Benuskova, & Kasabov, 2010), bioinformatics (Koefoed, Capecci, & Kasabov, 2018; Dray, Capecci, & Kasabov, 2018; Capecci, Lobo, Laña, Espinosa-Ramos, & Kasabov, 2019), and multisensory streaming data modelling (Tu, Kasabov, & Yang, 2016).

3.2 Information Encoding as Spikes

Spiking neural networks are inspired by the principles in the biological brain, where external information is encoded as short electrical pulses (Kasabov, 2019). The main principle is that real value of input information is optimally converted to spike events as a new form of input into SNN, preserving the task related information of the original signal during the encoding process.

Numerous spike encoding algorithms have been developed in the literature, some well-known algorithms are listed as follows, and one of the most popular encoding algorithms, called Threshold-based encoding, is discussed afterward.

- Threshold-based encoding (Delbruck, & Lichtsteiner, 2007)
- Rank Order Coding (Thorpe & Gautrais, 1998)
- Population Rank Coding (Bohte, 2004)
- Ben's Spike Encoding algorithm (Schrauwen & Van Campenhout, 2003)
- Step Forward Encoding algorithm (Kasabov, et al., 2016)
- Moving-Window Spike Encoding Algorithm (Kasabov, et al., 2016)

Threshold Based Encoding (TBR). This encoding method, a simple implementation of Temporal Contrast, was introduced by Delbruck and Lichtsteiner for the development of the Address-Event Representation (AER) system (Delbruck, & Lichtsteiner, 2007). In the method, changes between in signal amplitudes are compared with a given threshold, and a positive or negative spike is emitted according to whether the value exceeds or belows the encoding threshold. The

encoding threshold is given by a summation of the mean of the signal amplitude variation and its standard deviation multiplied by a factor, in which factor is a parameter of this encoding algorithm.

3.3 Computational Model of a Spiking Neuron

The structure of the biological neuron can be divided into three components: soma, dendrites and axon, as shown in Figure 3-1. The dendrites are positioned at the beginning to receive the electrical impulses from other neurons and transmits these signals to the soma. The cell body plays a key role in processing input spikes to maintain the function of a neuron. The output signal is delivered via axon to other neurons connected to it. The junction between two neurons is connected across synapses.

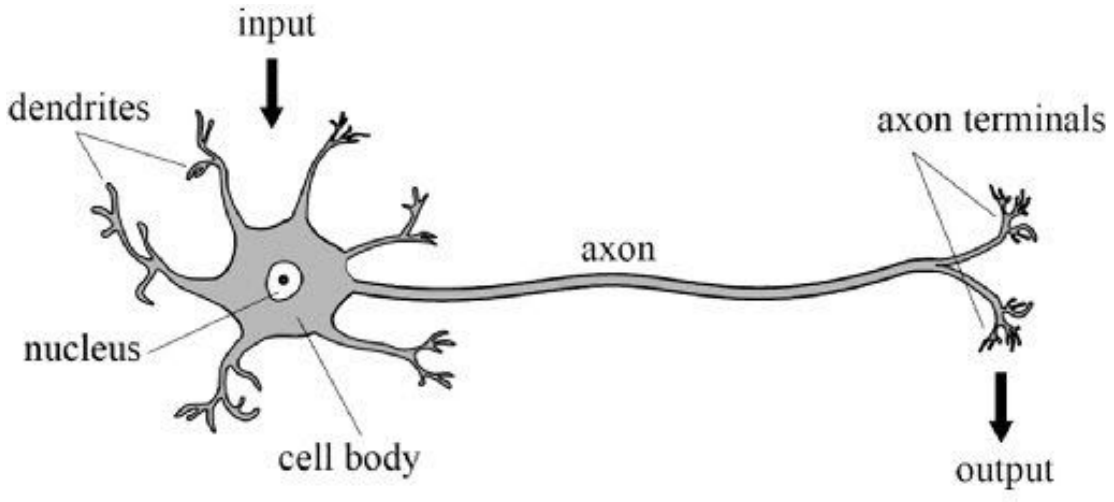

Figure 3-1 Structure of a biological neuron. Figure from (Neves, González, Leander, & Karoumi, 2017).

Different computational models of spiking neurons have been proposed in order to resemble a biological neuron. Some popular ones are: Hodgkin-Huxley Model (Hodgkin, & Huxley, 1952), Integrate-and-Fire Model (Abbott, 1999), Izhikevich Model (Izhikevich, 2003), Spike Response Model (Gerstner, & van Hemmen, 1992), Thorpe's Model (Thorpe, 1990), and Probabilistic and Stochastic Spiking Neuron Models (Kasabov, 2010). The main characteristics of Integrate-and-Fire Model is explained as follow:

The Leaky Integrate and Fire Model (LIFM): The integrate and fire neuron model was proposed by Lapicque (Abbott, 1999) and it is based on the principle of an electrical circuit consisting of a capacitor C in parallel with a resistor R to product current $I(t)$. The principal of LIFM can be conceived as a leaky integrator, where an action potential is emitted when membrane potential $u(t)$ reaches the critical voltage for spike initiation called threshold θ. After the arrival of a spike, the membrane potential decays back to the resting potential u_{rest}. The moment of threshold crossing defines the timing t at which a neuron fires. Before the next threshold crossing occurs, There is a refractory period during which the neuron cannot produce another action potential and the summation of the membrane potential slowly leaks over time, which is denoted by a temporal parameter τ. The structure and the functionality of the LIFM is illustrated in Figure 3-2 and the model can be described by the following differential equation:

$$\tau_m \frac{du}{dt} = -[u(t) - u_{rest}] + R\,I(t) \qquad (3\text{-}1)$$

Where the membrane time constant $\tau_m = RC$, u_{rest} is the resting potential, I is the input current, $u(t)$ is the membrane potential, and R is the resistance.

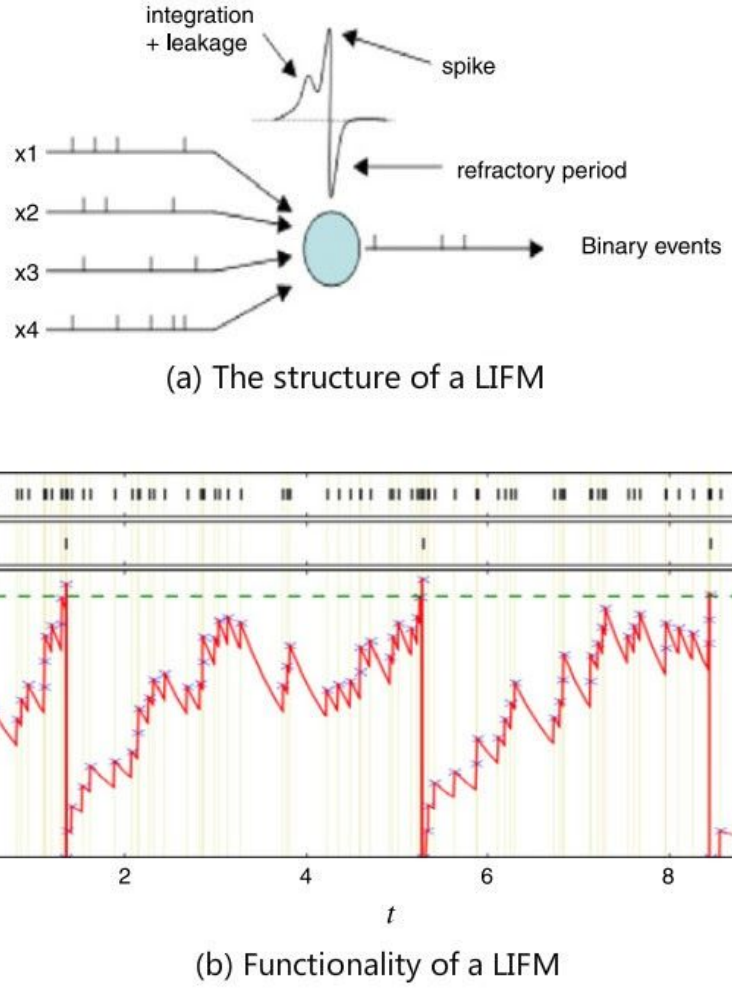

Figure 3-2 The LIFM of a spiking neuron. (a) Schematic representation; (b) Showing an input train of spikes (top row), the emitted output spikes (second row) and the membrane potential changes over time. Figure from (Kasabov, 2014).

3.4 Learning in SNN Models

Learning in SNN is the process of adjusting the connection weights between two spiking neurons. The most popular learning rules are SpikeProp (Bohte, Kok, &

Poutre, 2000), Spike-Time Dependent Plasticity (Song, Miller, & Abbott, 2000), Spike-Driven Synaptic Plasticity (Fusi, Annunziato, Badoni, Salamon, & Amit, 1999), Rank Order Learning Rule (Thorpe, & Gautrais, 1998). Two important learning rules are explained in the following:

Spike Timing Dependent Plasticity (STDP): This learning paradigm was inspired by the principle of Hebbian learning (Song, Miller, & Abbott, 2000; Hebb, 1949), in which the synaptic weights are adopted according to the temporal order of pre-synaptic and post-synaptic action potentials. In STDP learning, the arrival time of pre-synaptic spikes earlier than the post-synaptic spikes results in synaptic potentiation, namely Long-term Potentiation (LDP), while the timing of pre-synaptic spike activity after the post-synaptic spikes causes synaptic depression, namely Long-term Depression (LTD). The STDP learning rule is defined using the following equation:

$$W(t_{pre} - t_{post}) = \begin{cases} A_+ exp(\frac{t_{pre} - t_{post}}{\tau_+}), & \text{if } t_{pre} < t_{post}. \\ A_- exp(-\frac{t_{pre} - t_{post}}{\tau_-}), & \text{if } t_{pre} > t_{post}. \end{cases} \quad (3\text{-}2)$$

Where $W(t_{pre} - t_{post})$ defines the magnitude of this synaptic change based on the time interval ($t_{pre} - t_{post}$), as illustrated in Figure 3-3, the parameters A_+ and A_- refer to the learning rate and parameters τ_+ and τ_- define the time interval of pre-to-post-synaptic spike.

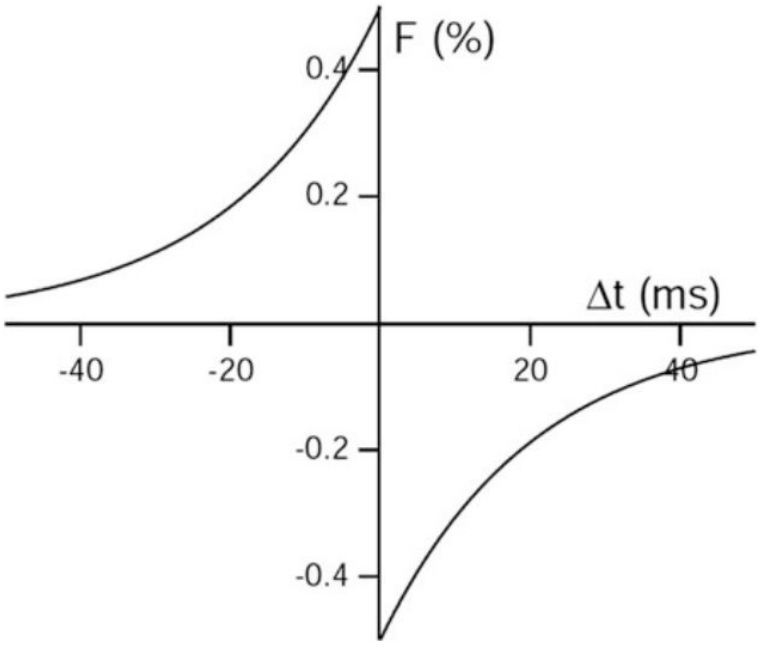

Figure 3-3 Synaptic change in a STDP learning neuron. Figure from (Song, Miller, & Abbott, 2000).

3.5 NeuCube - An SNN Framework

NeuCube is a Brain-Inspired Spiking Neural Network (BI-SNN) architecture and it has demonstrated the feasibility of learning from spatio-temporal data (Kasabov, 2014). The basic NeuCube model consists of the following five sub-modules, as illustrated in Figure 3-4.

- Input data encoding and mapping
- Unsupervised learning in a 3D SNN model
- Supervised Learning in an Evolving SNN

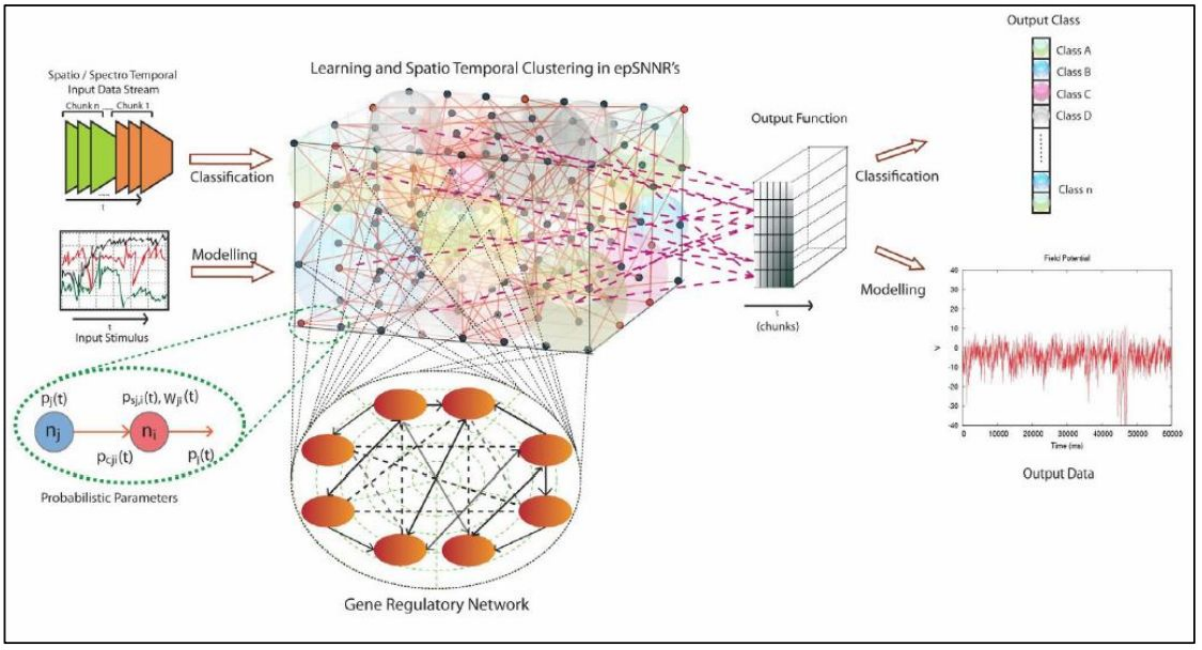

Figure 3-4 Block diagram of NeuCube architecture, including: input spatio-temporal data encoding module, 3D SNN module, output module for classification/regression, and gene regulatory network (GRN) module (optional). Figure from (Kasabov, 2012).

3.3.1 Input Data Encoding

The continuous time series of spatio-temporal data are first encoded into spike series, representing the time of changes in the input data. Algorithms for spike encoding implemented in NeuCube include:

- Threshold-based Encoding (Delbruck, & Lichtsteiner, 2007)
- Moving-Window Spike Encoding Algorithm (Kasabov, et al., 2016)
- Step Forward Encoding algorithm (Kasabov, et al., 2016)
- Ben's Spike Encoding algorithm (Schrauwen & Van Campenhout, 2003)

3.3.2 Input Data Mapping

After the encoding process, the encoded sequence of spikes are transferred into spatially located spiking neurons in the SNN model. In NeuCube, a recurrent 3D structure, namely SNNcube, is constructed to map the spatial components of input data according to a 3D brain template, such as Tailarach atlas (Talairach & Tournoux, 1988), Montreal Neurological Institute (MNI) template (Brett, Christoff, Cusack, & Lancaster, 2001) or other brain coordinate systems. After mapping input data spatially to the SNN model, the LIFM spiking neuron connectivity in the SNNcube is initialized using the small-world connectivity rule (Bullmore, & Sporns, 2009). Learning within the SNN model consists of two phases, including unsupervised and supervised learning. These learning processes are explained in Section 3.3.3 and Section 3.3.4 respectively.

3.3.3 Unsupervised Learning in SNNcube

The first stage is unsupervised learning that is intended to modify the neuronal connection weights in the SNNcube based on the learning of spatial and temporal associations from the encoded sequence of spikes. In the NeuCube framework, STDP learning rule (Song, Miller, & Abbott, 2000) is applied to the 3D SNNcube, resulting in an evolving connectionist structure in the SNNcube. In this learning, the arrival time of pre-synaptic spikes earlier than the post-synaptic spikes induces Long-term Potentiation (LDP), while the timing of pre-synaptic spike activity after the post-synaptic spikes causes Long-term Depression (LTD). The goal of the learning

rule is to capture the relative timing of pre-synaptic and post-synaptic activity, forming trajectories of connections to represent spatio-tempral patterns in the data.

3.3.4 Supervised Learning and Classification in Evolving SNN

Once the SNN model training in an unsupervised mode is completed, dynamic evolving SNN classifier (deSNN) (Kasabov, Dhoble, Nuntalid, & Indiveri, 2013) is performed in a supervised learning mode to learn the class labels relationship to each training sample. For each training sample, a new output neuron is assigned to the output neuronal layer, and all neurons from the SNNcube are connected to the output neurons in the SNN model. In the deSNN classifier, the initial connection weights of each synapse are established using the RO learning rule (Thorpe, & Gautrais, 1998) based on the first spike at this synapse. These connection weights need to be further adjusted through the SDSP algorithm (Fusi, Annunziato, Badoni, Salamon, & Amit, 1999). After that, this newly created neuron is added to the output neuron repository. Given a new spatio-temporal sample without class label information, the synaptic weights of a newly created output neuron are compared with the already existing output neurons in the output neuron repository during training. The class label of the new sample is determined by the closest output neuron using k nearest neighbors (KNN) method.

3.6 Chapter Summary

This chapter reviews computational models of SNNs and introduces the SNN framework NeuCube for modelling spatial-temporal data, enabling a better interpretation of spatio-temporal characteristics of data. In the next chapter, the proposed methods for transfer learning in BI-SNN will be introduced which is based on the NeuCube framework.

Chapter 4 Proposed Methodology for Transfer Learning in BI-SNN

4.1 Introduction

In the previous chapter, I have reviewed that the NeuCube framework is an ideal system for modelling complex spatio-temporal patterns. In this chapter, I proposed new adaptations based on the NeuCube framework for transfer learning scenarios, including incremental learning and transfer learning algorithms. In addition, I proposed three additional new algorithms, including two connection weights pruning algorithms and one neuron aggregation algorithm. These three algorithms can be integrated with the proposed transfer learning approaches.

4.2 Incremental Learning in BI-SNN

As transfer learning belongs to the class of incremental learning algorithms, here I have explored incremental learning in the deSNN. The incremental learning (ImSNN) algorithm is algorithmically described in Algorithm 1. In this method, samples from target tasks or subjects are presented one-by-one to the model without any data reinforcement from previous samples with the goal of learning target tasks or subjects sequentially. More specifically, the model first creates a repository of output neurons for the training patterns. When a new task or subject is given, each sample belonging to these new tasks or subjects will create a neuron to the output neuron repository sequentially. The learning procedure of the deSNN classifier relies mostly

on the repository of output neurons. Thus, the output neuron repository tends to store redundant information when processing target tasks or subjects. Here lies the rationale for merging similar neurons within the output neuron repository by applying the neuron aggregation technique, which is explored in the next subsection 4.5.

Algorithm 1 Proposed ImSNN algorithm

Input: Input spatio-temporal data

Parameters: encoding parameters;mod, $drift$, deSNN parameters;

1: Initialize output neuron repository, $NR = \{\}$,
2: **for** every input sample belonging to the learning task or subject **do**
3: Encode input data into spike sequences
4: deSNN classifier is performed to learn to classify/predict different dynamic patterns of the spike sequences
5: Add the output neuron to the output neurons repository NR
6: **end for**
7: Recall and test the model on all previously seen tasks or subjects
8: Go to 2 and repeat for subsequent tasks or subjects

Table 4-1 The proposed ImSNN algorithm for deSNN (Kasabov et al. 2013).

4.3 Proposed Transfer Learning Algorithm for BI-SNN

Algorithm 2 reflects the adaptations of the SNNcube in the original NeuCube architecture in order to deploy it in transfer learning scenarios, namely TrSNN. The training procedure is similar to ImSNN approach Algorithm 1. The only difference between the proposed TrSNN approach (Algorithm 2) and the ImSNN approach (Algorithm 1) is that TrSNN performs both unsupervised learning and supervised learning steps, while ImSNN uses deSNN classifier only. Given an existing model trained with the source tasks or subjects, when a new task or subject is given, a new

SNNcube is created, and the neural connectivity with the target tasks or subjects are learned based on the prevised trained SNNcube. At the supervised learning stage, each target sample will create an output neuron to the output neuron repository sequentially. The process of unsupervised learning can be regarded as a learning process that forms deep patterns of connectivity between individual neurons based on the timing of their spiking activity and the connection between these neurons. Furthermore, the knowledge stored in the SNNcube can be transferred to the subsequent trained models such that subsequent tasks or subjects can reuse this common knowledge to improve the learning performance. This leads to a more suitable model design for transfer learning environments. Thus, TrSNN has the capacity to share and reuse this common knowledge.

Algorithm 2 Proposed TrSNN algorithm

Input: Input spatio-temporal data

Parameters: encoding parameters; SNNcube parameters; mod, $drift$, deSNN parameters;

1: Initialise a 3D SNNcube structure with input neurons using a suitable initialisation of initial weights
2: Initialize output neuron repository, $NR = \{\}$,
3: **for** every input sample belonging to the learning task or subject **do**
4: Encode input data into spike sequences
5: Unsupervised learning is performed in the SNNcube using STDP learning rules
6: deSNN classifier is performed to learn to classify/predict different dynamic patterns of the SNNcube activities
7: Add the output neuron to the output neurons repository NR
8: **end for**
9: Recall and test the model on all previously seen tasks or subjects
10: Go to 3 and repeat for subsequent tasks or subjects

Table 4-2 The proposed TrSNN algorithm for NeuCube.

4.4 Connection Weights Pruning Techniques for TrSNN Models

The TrSNN approach sketched in Algorithm 1 is the basis for our proposed approaches hybridized with connection weights pruning schemes, in which SNNcube pruning and output layer pruning techniques are applied on unsupervised learning stage and supervised learning stage respectively. The resulting hybrid approaches are labeled as TrSNN-CP and TrSNN-OP. In these connection weights pruning schemes, only the informative connections from the SNNcube to the corresponding output neurons in the classifier are left, and the others are removed. By cutting off certain connections in the network, such that new knowledge can take advantage of previously learned features but cause no interference in the pathways of the previously learned samples. The main goal is to forget outlier or stale information rather than the similar trajectories that may be essential for previous knowledge, ensuring commonly-used or recent information are retained.

4.4.1 SNNcube Pruning

In this method, inactive neurons in the SNNcube, which did not have active connections with other neurons, were suspended. The SNNcube pruning performed in the unsupervised learning stage is algorithmically described in Algorithm 3. Given a NeuCube learned from the previous task or subject, the connection weight of the learned SNNcube can be pruned before feeding the subsequent tasks or subjects into the NeuCube model. The pruned thresholds are adaptive based on the minimum

and maximum value of the weights, such that the threshold can be dynamically changed instead of fixed thresholds. After that, the pruned weights are reset to initial values, giving a chance to the subsequent tasks or subjects for continued training.

Algorithm 3 SNNcube pruning algorithm

Input: connection weights in the SNNcube $w_{j,i}$ from previous task or subject; initial SNNcube connection weights w_{init}

Parameters: P, Pruning percentage

Output: pruned connection weights

1: Get the number of pruned connection weight $N_{pos} = count(w_{pos}) * P$, $N_{neg} = count(w_{neg}) * P$

2: Get a list of potential thresholds based on the minimum and maximum value of the weights connectivity, T

3: **for** every threshold θ in the threshold list **do**

4: **if** the number of $prune\_synapses$ less than N_{pos} and N_{neg} **then**

5: $\theta_{pos} = \theta$ or $\theta_{neg} = \theta$

6: Go to 9

7: **end if**

8: **end for**

9: Get the prune synapses using $j = f(w_{j,i} : \forall j, \theta),\ j \in prune\_synapses$ and $f(x) = \begin{cases} j, & \text{if } x \geq \theta_{pos}, \& \, x < \theta_{neg} \\ \{\}, & \text{otherwise.} \end{cases}$

10: Set the prune synapses to initial values, $w[prune\_synapses] = w_{init}[prune\_synapses]$

Table 4-3 The proposed SNNcube pruning algorithm.

4.4.2 Output Layer Pruning in the deSNN

The output layer weight pruning method consists of two parts. First, during the training of each task, a weight regulator is added to promote sparsity in the output layer and to regulate the density of connections from the SNNcube to the corresponding output neurons. The second part of the sparsification scheme is

pre-validating weight pruning based on the connection state of each training sample, as each training sample has a different pruning position.

A precise definition of the pruning strategy is as follows. Given an output layer in NeuCube, composed of N_i neurons, each has been trained from different training samples. In order to find the active and inactive connection weights between SNNcube and output neurons, the minimum weight was computed for each individual output neuron. Connection weights below a threshold are considered critical, while others are non-critical. The weights of all non-critical synapses are reduced to zero.The threshold value θ is a post-training hyperparameter, which can control over the amount of sparsity in the output layer. Furthermore, if $\theta = 0$, the function being computed is entirely by the dense network. The weight pruning strategy is algorithmically described in Algorithm 4.

The validation phase is carried out by first pruning away the unused connections of the newly created output neuron based on the pruning index of all trained output neurons that are within the same output class group in the repository. The class label for the validation sample is assigned according to synaptic similarity of the newly pruned output neuron and the already trained output neurons.

Algorithm 4 deSNN algorithm with Output Layer Pruning

Input: The output of SNNcube or input spatio-temporal data
Parameters: mod, $drift$, deSNN parameters; θ, Pruning interval
Output: NR, output layer neurons

1: Initialize output neuron repository, $NR = \{\}$
2: **for** every samples **do**
3: Create a new output neuron i and the connection weights as $w_{j,i}$ using RO and SDSP learning rule, where j is the pre-synaptic neuron of the output neuron i
4: Get the minimum connection weights of output neuron i, min_i
5: Get the prune synapses using $j = f(w_{j,i} : \forall j, \theta)$, $j \in prune\_synapses$ and $f(x) = \begin{cases} j, & \text{if } x < min_i + \theta \\ \{\}, & \text{otherwise.} \end{cases}$
6: Set the synapses to 0, $w[prune\_synapses] = 0$
7: Add the new pruned output neuron to the output neurons repository NR
8: **end for**

Table 4-4 The proposed output layer pruning algorithm for deSNN (Kasabov et al. 2013).

4.5 Neuron Aggregation Technique for TrSNN Models

The proposed TrSNN approach sketched in Algorithm 1 can also be combined with a neuron aggregation technique, in which the output neurons which belong to the same class and have similar connection weights are merged in an incremental way through averaging to obtain a compact feature representation and in this way the structure of network can evolves continuously in a life-long manner. The neuron aggregation performed in the deSNN classifier is algorithmically described in Algorithm 5.

Given an output layer in NeuCube, composed of a repository of output neurons, each has been trained from different training samples. For every new training sample, a new output neuron is allocated and connected to all neurons in the SNNcube through $w_{j,i}$. The weight matrices of the newly produced output neuron and the already trained output neurons in the repository are then compared. If the new neuron weight vector is similar to any of the any of the already trained output neurons using Euclidean distance, these two neurons will be merged by averaging the connection weights using following equation:

$$w_{j,i} = \frac{w_{new} + w_{j,i}}{M} \qquad (4\text{-}1)$$

Where M is the number of neurons being aggregated on the output layer. After merging the neurons, the new created output neuron will be discarded. If there are no existing trained neurons in the repository found to be similar to the new output neuron, then it will be added to the corresponding class pool of neurons in the repository. One post-training hyperparameter was introduced to control its similarity with the other output neurons during training, namely similarity parameter SIM. Furthermore, if SIM = 0, the function being computed is entirely by the model without neuron aggregation.

Neuron aggregation can be also used along with connection weights pruning algorithms. The flow diagrams for the TrSNN model with weight connection pruning methods and neuron aggregation are presented Figure 4-1 and Figure 4-2 respectively, namely TrSNN-CP-NG and TrSNN-OP-NG. Notice that when neuron aggregation is used with output layer pruning, repruning the connection weights is required for the merged neurons.

Algorithm 5 deSNN algorithm with Neuron aggregation algorithm

Input: The output of SNNcube or input spatio-temporal data
Parameters: SIM, similarity parameter
Output: NR, output layer neurons

1: Initialize output neuron repository, $NR = \{\}$
2: **for** every samples belonging to class c **do**
3: Create a new output neuron i and the connection weights as $w_{j,i}$ using RO and SDSP learning rule, where j is the pre-synaptic neuron of the output neuron i
4: **if** min distance(Newly output neuron weight vector, Neuron repository weight vectors within the same output class group in NR) $\leq SIM$ **then**
5: Update the weight vector of the most similar neuron $w_{j,i} = \frac{w_{new}+w_{j,i}}{M}$, where M is the number of neurons being aggregated
6: **else**
7: Add the new output neuron to the output neurons repository NR
8: **end if**
9: **end for**
10: Go to 2 and repeat for all target classes

Table 4-5 The proposed neuron aggregation algorithm for deSNN (Kasabov et al, 2013).

.

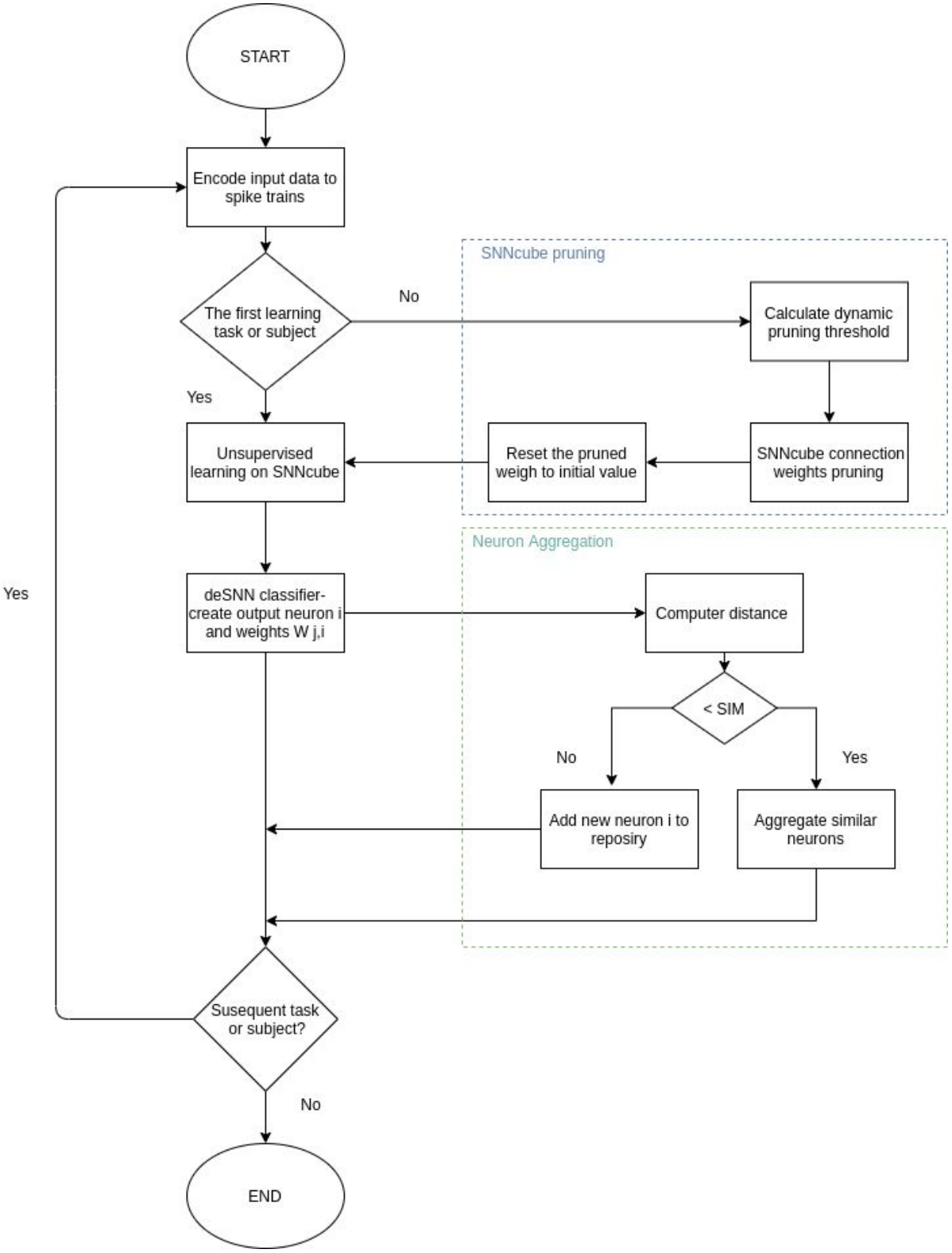

Figure 4-1 Schematic diagram of the proposed TrSNN-CP-NG.

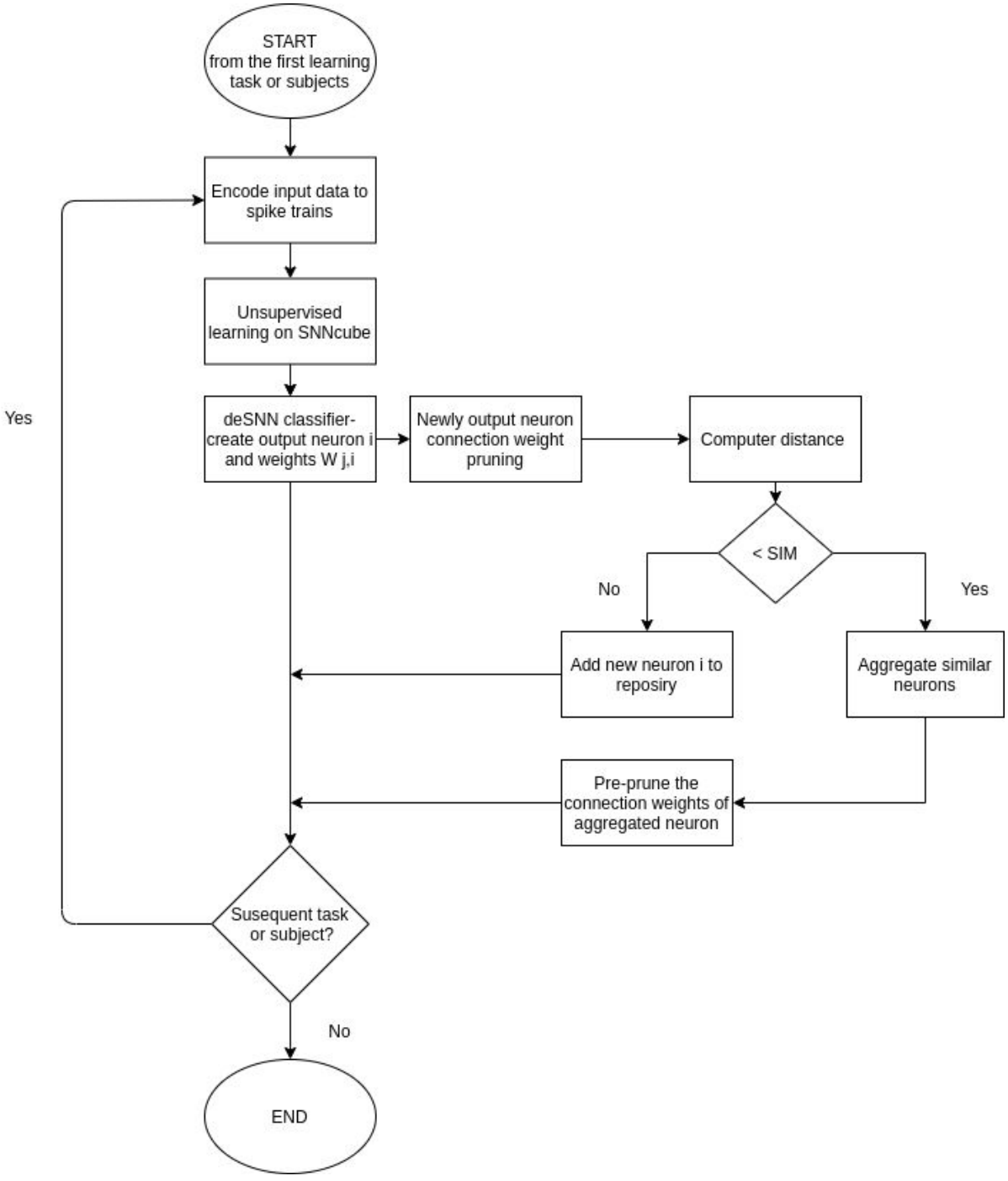

Figure 4-2 Schematic diagram of the proposed TrSNN-OP-NG.

4.6 Chapter Summary

In this chapter, a methodology and algorithms are proposed for transfer learning in the NeuCube architecture through introducing new methods for learning of

sequentially presented tasks and subjects, resulting in the ImSNN and TrSNN algorithms. Three techniques, including two connection weights pruning algorithms and one neuron aggregation algorithm, were also introduced as an additional algorithm that can be freely cooperated with the TrSNN learning procedure. In the next two chapters, the proposed methods will be applied to extensive experiments with the aim of evaluating the performance of the proposed schemes on a real-life case study of EEG data from two empirical studies, including transfer learning across tasks in one subject and transfer learning across subjects in one task.

4.7 Contribution

In this chapter, I have made the following original contributions:

- Proposal of an incremental learning approach based on the SNN architecture, called ImSNN
- Proposal of a transfer learning approach based on the SNN architecture, called TrSNN
- Proposal of two connection weights pruning techniques for TrSNN Models, namely TrSNN-CP and TrSNN-OP
- Proposal of a neuron aggregation techinque for TrSNN Models

Chapter 5 Modelling Transfer Learning of New Tasks by a Single Subject in NeuCube

5.1 Introduction

This chapter applies the proposed transfer learning methods (introduced in Chapter 4) to a case study of EEG data based on the scenario of task-to-task transfer learning in one subject. In this study, I designed a series of experiments to examine the effect of transfer learning approaches on different functional modules of NeuCube architecture. I constructed optimal SNN models and trained them with an EEG dataset related to functional upper limb movements. The models are also evaluated in a three-phase analysis, including SNNcube patterns in an unsupervised mode, output layer patterns in a supervised mode, and the final experiment results to evaluate the transfer learning performance.

5.2 Dataset and Preprocessing

The functional upper limb movements dataset was used in this case study (Mohseni, Shalchyan, Jochumsen, & Niazi, 2020). This dataset was collected by New Zealand College of Chiropractic and Aalborg University under the ethical approval of the local ethics committee (N-20130081), and it consists of EEG data from 12 healthy subjects by 64 EEG channels at 512 Hz. Each subject was instructed to perform four classes of motor imagery tasks as follow:

- Class 1: Reach for a glass of water, drink and place the glass on the table
- Class 2: Throw a ball from the right hand to the left hand
- Class 3: Lift a tray from the table and place the tray on the table again
- Class 4: Push a glass from position A to position B

Classes 1, and 4 are unilateral movements, while classes 2 and 3 are bilateral movements. As shown in Figure 5-1, at the beginning of a trial, the subjects were instructed to hit the ‘s’ key on the keyboard before executing the movements. When the subject had finished the task, the experimenter hit the ‘s’ key again. In this period, the subject was asked to perform a specific task. Then the visual cue disappeared from the screen, and a short break followed until the next trial began. Each subject had 50 trials from each class. Table 5-1 summaries the characteristics of the dataset.

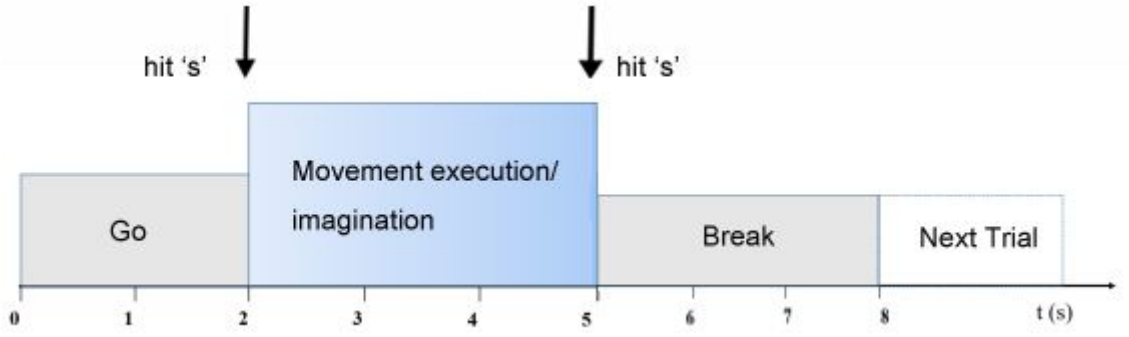

Figure 5-1 Timing scheme of the motor imagery tasks.

Data preprocessing was performed using the Matlab EEGLAB toolbox (Delorme, & Makeig, 2004). The continuous EEG data was first extracted between [0.5, 2.5] seconds after the start of each trial (odd index values of ‘s’). Next the EEG signals

were downsampled from 512 Hz to 256 Hz. A butterworth band-pass filter (2the order between [0.05, 45] Hz) and 2th order notch Filter with a lower cutoff frequency of 49 Hz and a higher cutoff frequency of 51 Hz were then applied to the data. Finally, an EEGLab plug-in ADJUST was used to remove the artifacts. Figure 5-2 shows the topological graph for EEG channels. The spatial mapping of EEG data in the Talairach space (Talairach & Tournoux, 1988) is presented in Appendix A.

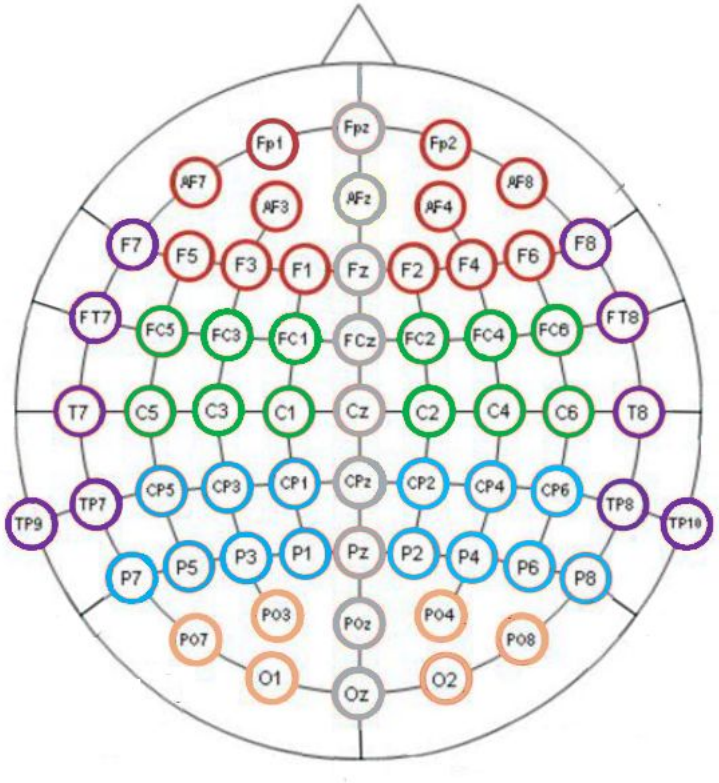

Figure 5-2 Topological graph for EEG channels in 10-20 standard.

Table 5-1 Summary of the upper limb movements dataset.

	Number of				
	EEG Channels	Subjects	Time samples	Classes	Trials/class/subject
Functional upper limb movements dataset	64	12	768	4	50

5.3 Experimental Design

To validate the efficacy of the proposed approaches in transfer learning across tasks in one subject, the NeuCube SNNcube and deSNN classifier were first trained with samples from source task, class 1. After that, samples from other three classes/target tasks were presented one-by-one to and learned without requiring a retraining mechanism from previous classes with the goal of transferring some knowledge acquired from the source tasks to the target tasks. To examine the effect of transfer learning on different functional modules of NeuCube architecture, a series of experiments were conducted to validate the performance of the proposed approaches. Table 5-2 summarises the details of all experiments. A baseline model which trains in a batch mode on all data simultaneously was established as a baseline for comparison and both transfer learning model and baseline model were performed using the same initial connection weights.

Table 5-2 Description of experiment schemes.

	Description
Baseline	batch learning of all tasks using NeuCube
ImSNN	Incremental learning of each task sequentially on deSNN classifier only
TrSNN	Transfer learning of each task sequentially in unsupervised mode and deSNN classifier
TrSNN-CP	Transfer learning of each task sequentially in unsupervised mode and deSNN classifier, along with weight connection pruning in the SNNcube after learning of each task
TrSNN-CP-NG	Transfer learning of each task sequentially in unsupervised mode and deSNN classifier, along with weight connection pruning in the SNNcube and neuron aggregation in the deSNN classifier

TrSNN-OP	Transfer learning of each task sequentially in unsupervised mode and deSNN classifier, along with connection weight pruning between SNNcube and the output neurons in the classifier
TrSNN-OP-NG	Transfer learning of each task sequentially in unsupervised mode and deSNN classifier, along with connection weight pruning between SNNcube and the output neurons in the classifier and neuron aggregation in the deSNN classifier

The flowchart of preparation of training and testing datasets for different schemes is presented in Figure 5-3. The encoding procedure to transform input signals into spike trains was a threshold-based encoding algorithm as explained in Chapter 3 and a fixed threshold was performed on each incoming task.

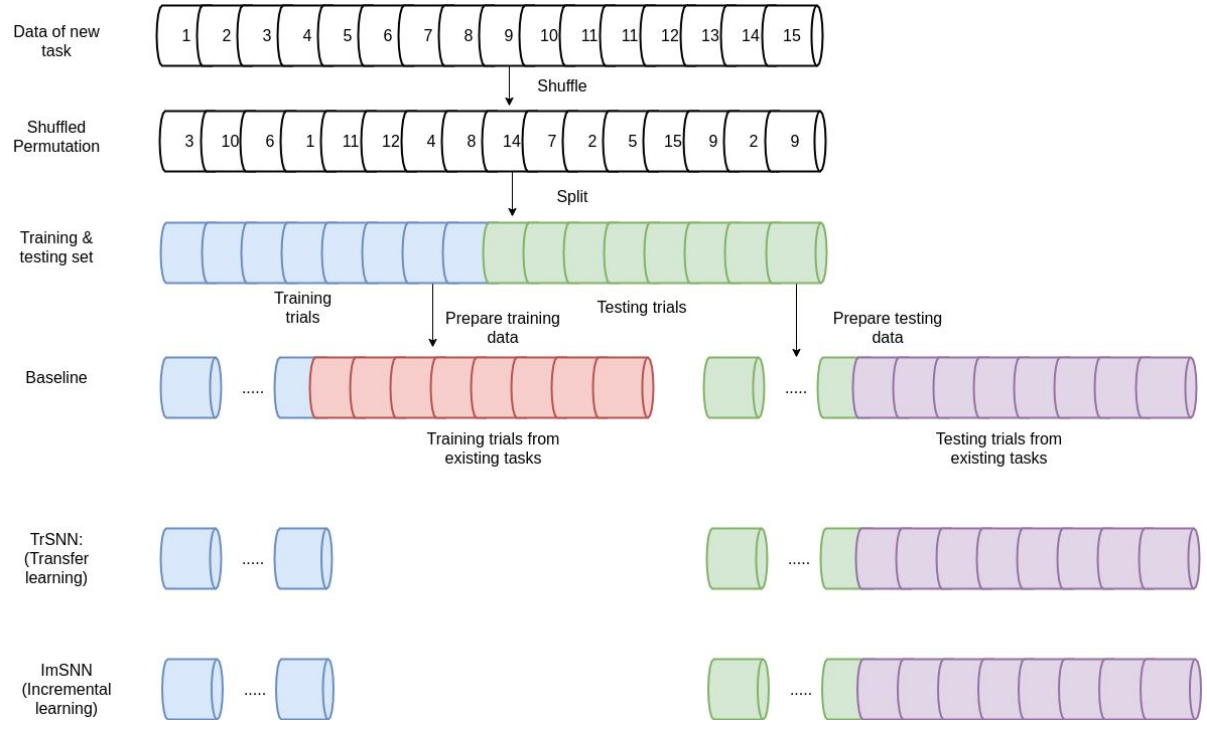

Figure 5-3 The flowchart of preparation of training and testing datasets for task-to-task transfer learning in one subject scenarios.

To demonstrate the performance of transfer learning over time, the NeuCube models were saved at each task change and the testing results of all previously seen tasks were evaluated at each task change. All experiments were performed five times and reported the test accuracy and its standard deviation on all tasks.

The parameter optimization phase is carried out on TrSNN models only. Parameters Drift, Mod, pruning percentage p, and pruning threshold θ were chosen using grid search and picked the value of the best 3 fold cross validation accuracy across all hyperparameters. For similarity parameter (SIM) in the neuronal aggregation procedure of the deSNN classifier, a slightly different optimization scheme was used. The models were first trained over a range of similarity parameters along with the hyperparameters optimized as above using grid search. The optimal similarity parameter (SIM) was chosen as the value of the best training accuracy to make sure the performance of the neuron aggregation scheme performs on par with the model trained without neuron aggregation on the training dataset. Furthermore, baseline and ImSNN models were also conducted with the default parameters for comparison. The details of the default parameters are presented in Appendix B, Section B-1. For TrSNN experiments, there are some additional optimized parameters, which is summarized in Table 5-3.

Table 5-3 Parameter settings for each TrSNN experiment.

	Parameters
TrSNN-CP	Drift: 0.005 Mod: 0.8 SNNcube pruning percentage: 0.995
TrSNN-CP-NG	Drift: 0.005

	Mod: 0.8 SNNcube pruning percentage: 0.995 SIM parameter: 2
TrSNN-OP	Drift: 0.005 Mod: 0.8 Output layer pruning threshold: 0.533
TrSNN-OP-NG	Drift: 0.005 Mod: 0.8 Output layer pruning threshold: 0.533 SIM parameter: 2

5.4 Experimental Results

The experimental results were organised in a three-phase analysis as follows:

- Analysis of SNNcube patterns in an unsupervised mode
- Analysis of output layer patterns in a supervised mode
- Analysis of the results to evaluate the transfer learning performance

5.4.1 Analysis of SNNcube Patterns in an Unsupervised Mode

After the unsupervised learning stage, the inactive connection weights in SNNcube which did not have active connections with other neurons, were temporarily pruned, allowing the model take advantage of previously learned features when learning new tasks, but cause no interference in the pathways of the previously learned tasks. Figure 5-4 (b) and (d) show the weights connectivity of the complete and pruned SNNcube after learning all four disjoint tasks sequentially. Further quantitative information of the complete and pruned SNNcube are plotted in Figure 5-4 (a) and (c), respectively.

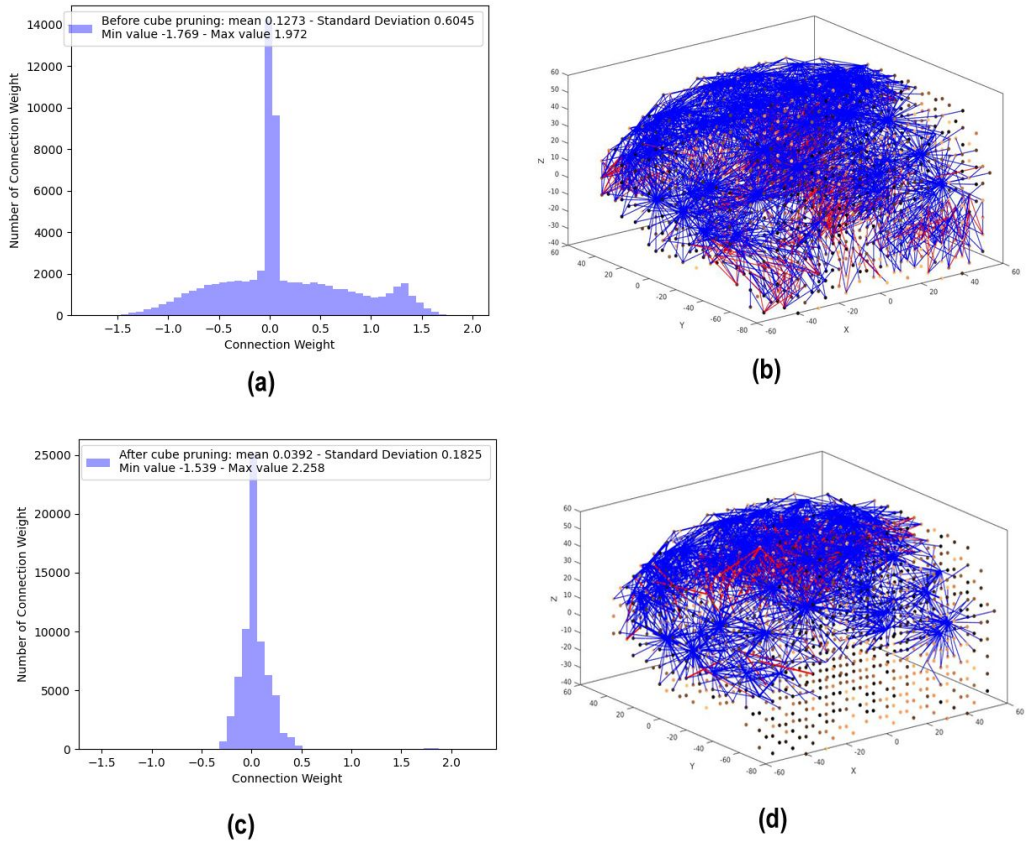

Figure 5-4 The distribution of connection weights for TrSNN models before and after SNNcube pruning.

The weights connectivity of SNNcube after connection weights pruning at each stage of the learning process are visualized in Figure 5-5 (a)-(d), and It can be seen from Figure 5-5 (a)-(d) that stronger connectivity is observed with further trained SNNcube. To perform a better analysis of EEG changes at each task change, the differences between the SNNcube for each stage of the learning process were computed through subtracting with the previous trained cube, which allows visualising the changes in neural connectivity as a result of transfer learning over time. In the graphs shown in Figure 5-5 (e), (f), and (g), further trained SNNcube resulted in a similar pattern of changes in some regional activation across all the three new tasks. However, the size of the activated connectivity was higher in

SNNcube trained with class 3, compared to class 2 and 4. The connection weight varied to different degrees as new tasks were added, demonstrating stronger activity throughout the entire training process. Further analysis of the connectivity patterns of SNNcube can be performed in several ways, such as quantitative analysis. Another more efficient way to analyse the learning patterns is through deep knowledge representation, which will be discussed in Chapter 7.

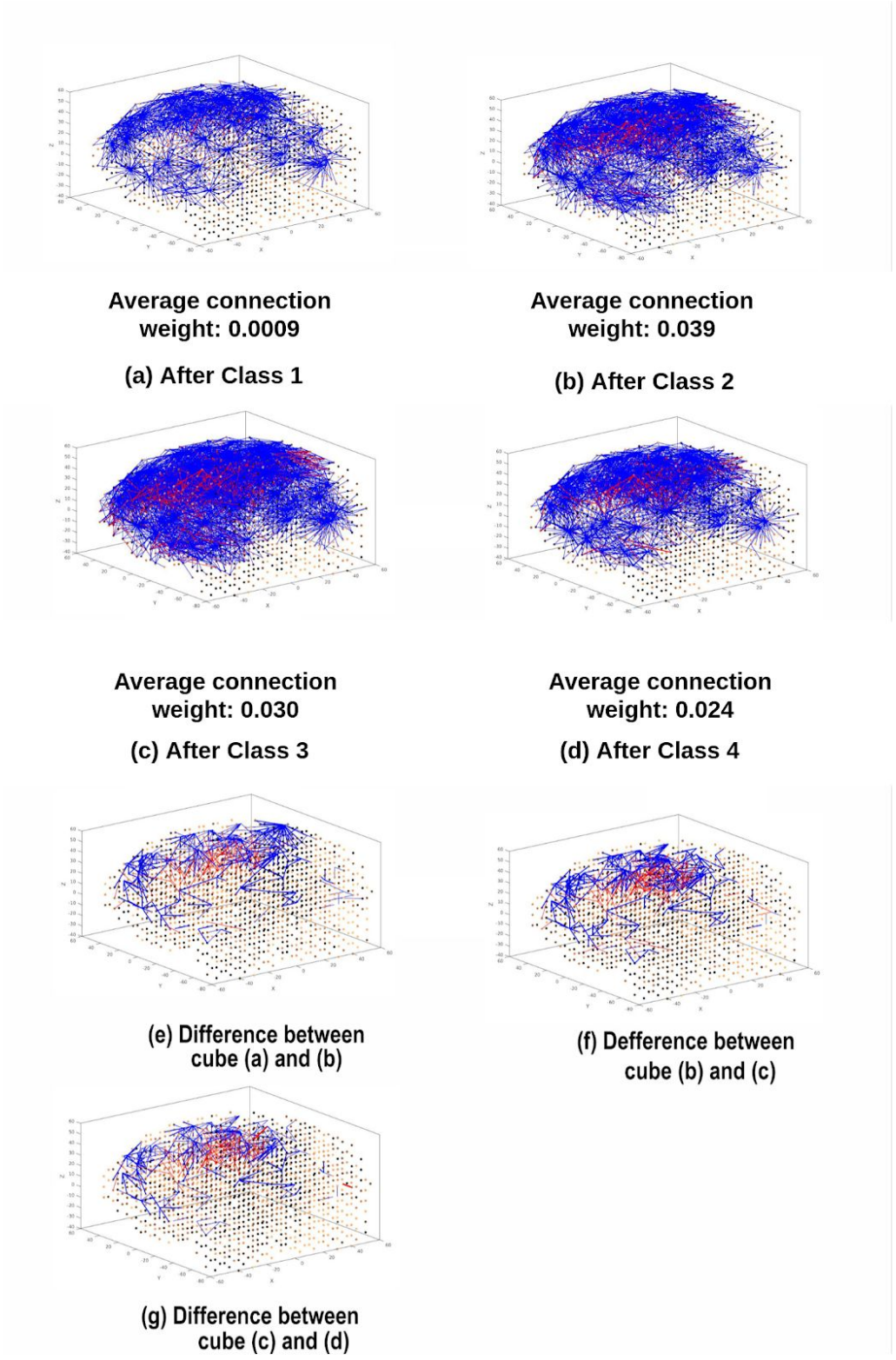

Figure 5-5 The connection weights of the pruned SNNcube for TrSNN models trained; (a) after class 1; (b) after class 2; (c) after class 3; (d) after class 4. Differences between the connectivity in the sequentially trained TrSNN models are shown in figs (e),(f),(g). The more new classes are added, the less new connections are added, as for the classification of new classes data, some of the previously created connections are utilised.

In order to analyse the information interaction between the brain areas after each new task, the total temporal interactions in terms of spike communication between input neurons was depicted. As illustrated by the Feature Interaction Network graph in Figure 5-6, thicker interaction lines were formed between the EEG channels positioned at Parietal region for the model trained with class 1 when compared with the after class 2, class 3, and class 4 groups in Figure 5-6 (b, c, d).

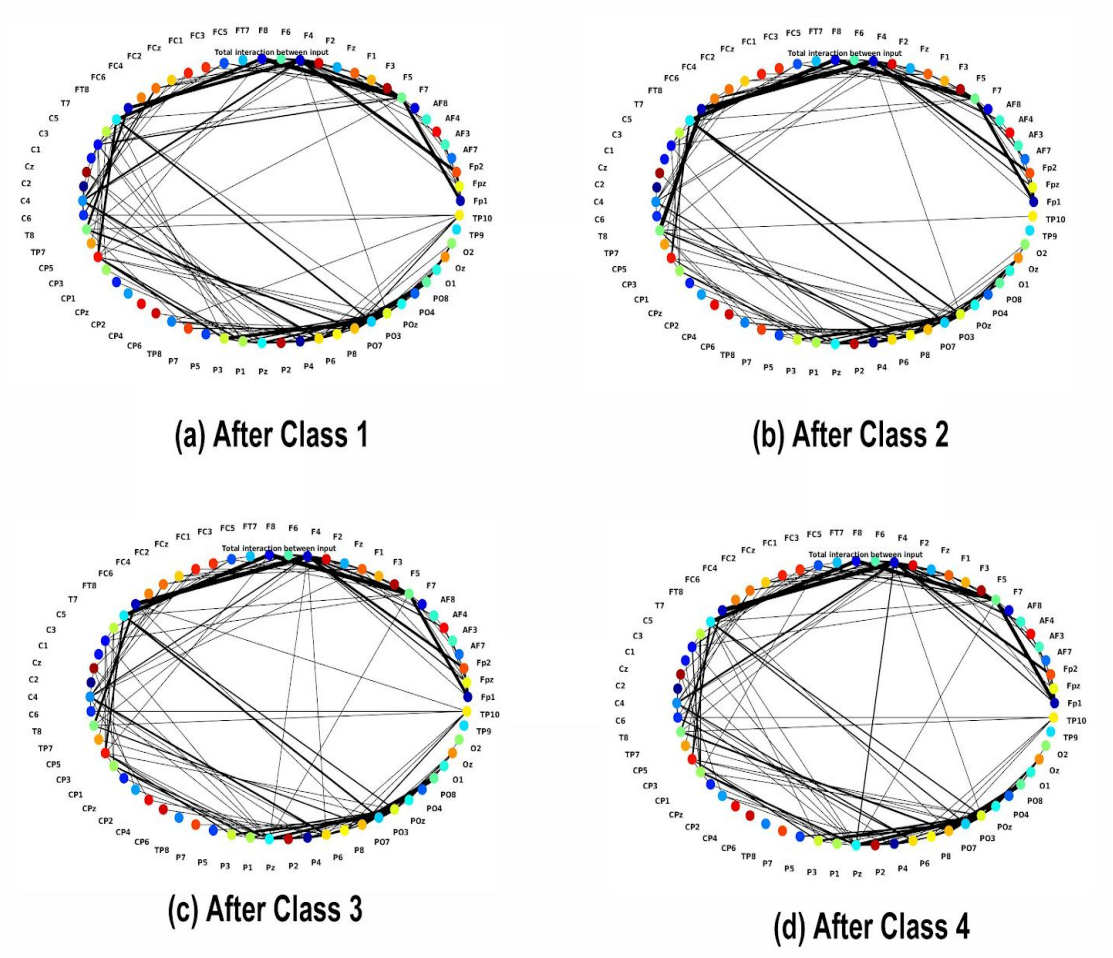

Figure 5-6 The Feature Interaction Network (FIN) captured the total spike interaction between the areas in TrSNN models representing 62 EEG channels as input neurons during the STDP learning at each stage of the learning process for the pruned SNNcube.

5.4.2 Analysis of Output Layer Patterns in a Supervised Mode

As mentioned in Chapter 3, all neurons in the SNNcube are connected to every neuron in the output layer. Therefore, after output layer pruning, inactive neurons in the output layer, which did not have activate connection with SNNcube, were pruned, which reduced the output connections for all samples from 93513 to 73875, as shown in Table 5-4, a significant reduction of the dimensionality of the space for classification.

Table 5-4 The number of connections of the output layer for the TrSNN model before and after pruning.

Output Layer Pruning		
Model	**Total**	**Density**
Complete	93513	100%
Pruned	73875	79%

The number of output neurons before and after aggregation is illustrated in Appendix B, Figures B-2.

5.4.3 Analysis of the Results on Test Data to Evaluate the Transfer Learning Performance

Figure 5-7 shows the final test results of each individual task for all experiments by the end of the training process and compares these results to the accuracy of the baseline model that allows data reinforcement from previous classes throughout the

training process. As shown in Figure 5-7 that the TrSNN model showed a clear age dependent decline in which older tasks were solved with lower accuracy (Figure 5-7; green). This penalty is expected due to sequentially training the tasks without knowledge of a task change. One possible reason is that all neurons from the SNNcube are connected to every output neuron, including old and new, and the new patterns in the SNNcube learned for the newly introduced task intervene with the old ones. When enabling weight pruning methods, with a sensible choice for threshold, even with the same penalty, the models outperformed the previous results by a noticeable margin (Figure 5-7; red and purple vs green). In fact, even with this penalty, the proposed TrSNN-CP achieved a respectable 81% test accuracy after transfer learning, which performed on par with the baseline systems, averaging 3% accuracy degradation. This suggests that given a sparse network trained on a number of tasks sequentially, it reduces the degradation of accuracy significantly for older tasks while learning new tasks. When comparing the performance of networks which had trained with neuron aggregation (Figure 5-7; red vs purple, and purple vs pink), these models yielded similar test accuracies. The ImSNN model roughly achieved a 91% test accuracy, resulting in the highest performance among all models. However, this finding was unexpected because new connections from new tasks are created in the SNNcube in TrSNN models and they were expected to obtain better performance. While the incremental learning in the deSNN demonstrated higher accuracy of classification, this type of learning does not transfer knowledge from one class to another, but simply creates new output neurons for new classes, separate from the previously created for previously learned classes. Further, the final F-Score for all experiments are presented in Figure 5-8.

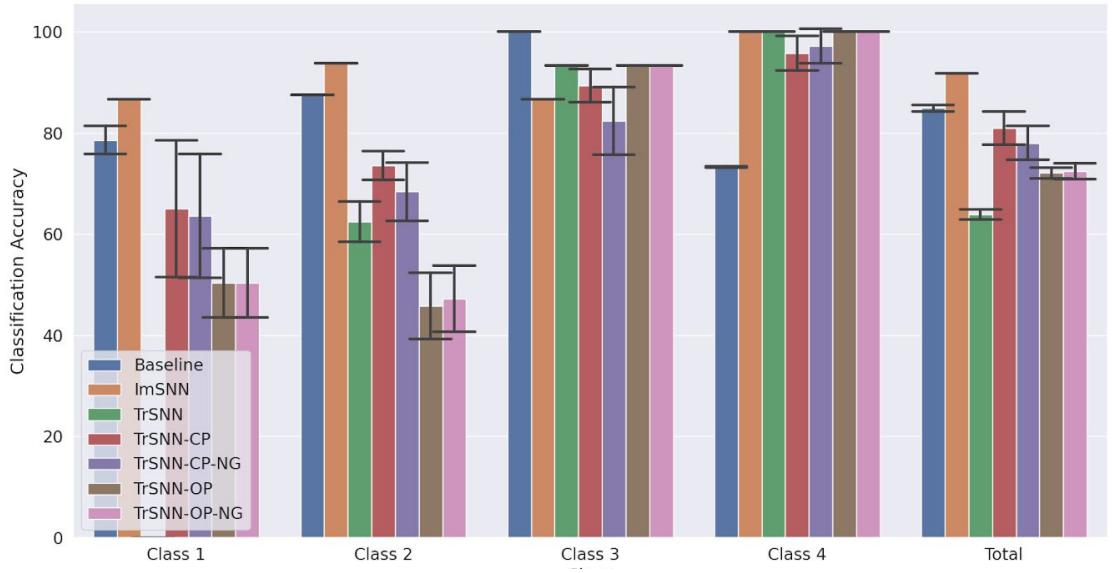

Figure 5-7 Final per-task accuracy for each experiment after training four classes with one single subject.

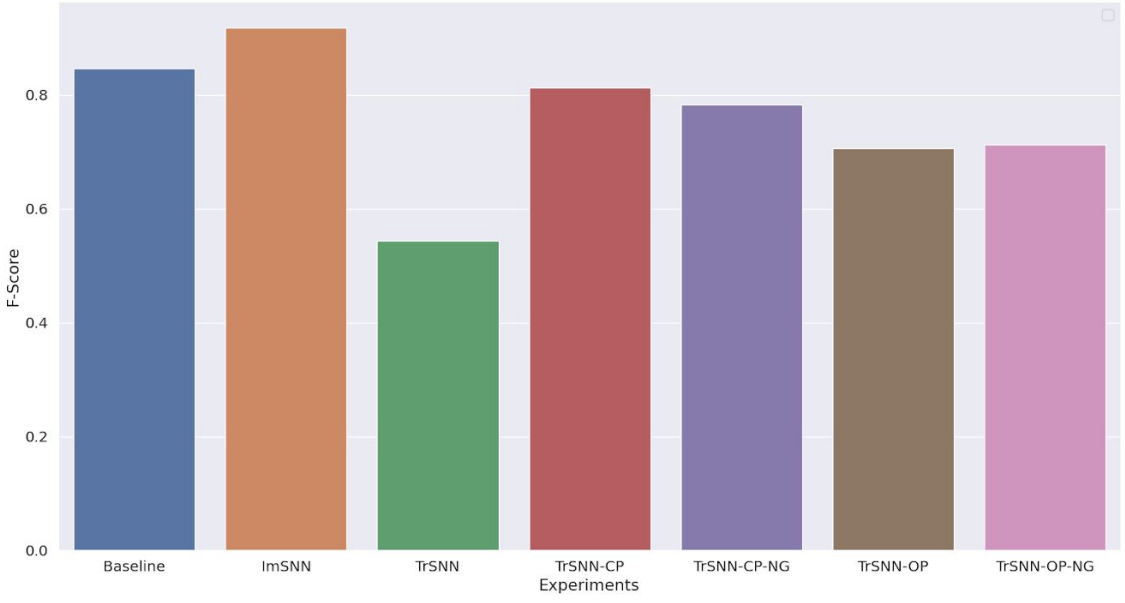

Figure 5-8 Final F-score for each experiment after training four classes with one single subject.

Besides the final accuracies of the transfer learning system, the performance throughout the training process is also depicted in Figure 5-9. As shown in Figure

5-9, graceful degradation of total accuracies when adding more training tasks were reduced significantly with connection weights pruning and neuron aggregation approaches.

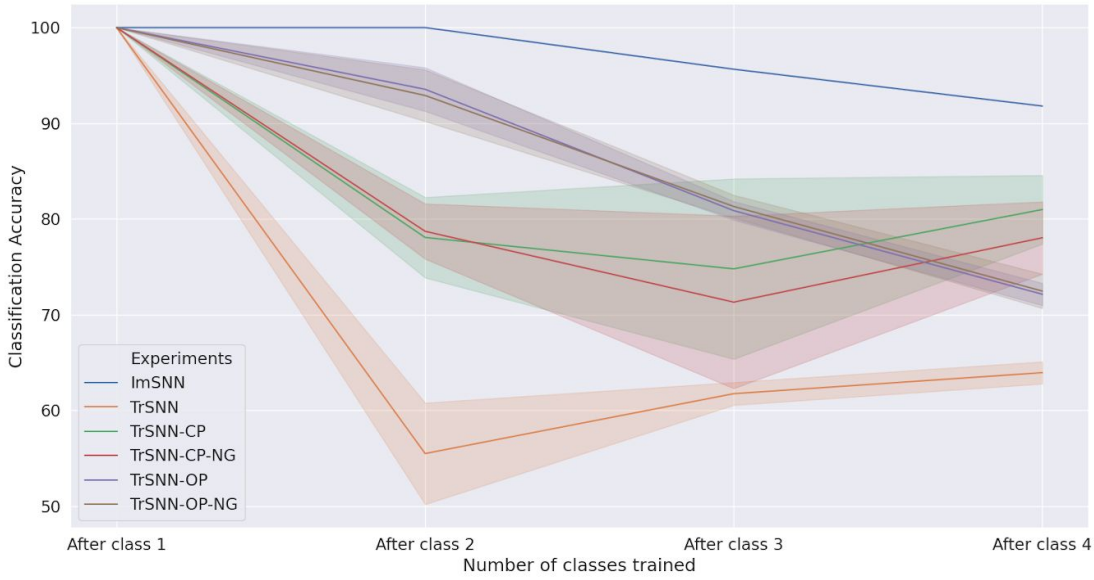

Figure 5-9 Per-task classification accuracy as new tasks are added over time, up to and including the current task for all experiments.

To gain further understanding of the types of errors that occur during the testing stage, the confusion matrices of the classification results trained with the TrSNN-CP-NG model is plotted in Figure 5-10. Notice that for this experiment, a large number of class 1 were misclassified as class 4. This can be explained by the large interclass ambiguity between class 1 and 4. These two classes exhibited high similarity with the areas of neurons in the SNNcube for the activation of this output neuron, hence the misclassification was mainly among the class 1 and class 4.

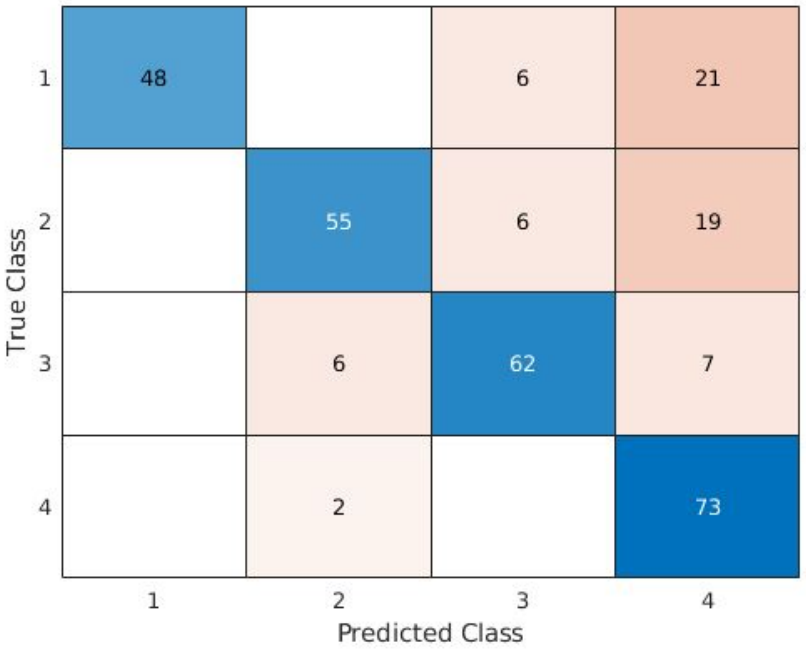

Figure 5-10 Confusion matrices of the classification results trained for TrSNN-CP-NG model.

5.5 Chapter Summary

In this chapter, I presented a case study of transfer learning across tasks in one subject based on the NeuCube SNN architecture using EEG data. Some of the key findings of this chapter are as follows:

1. The proposed methods for transfer learning in NeuCube make it learn new patterns through making new spatio-temporal trajectories for new tasks.

2. All TrSNN variants (TrSNN-OP, TrSNN-CP, TrSNN-OP-NG, TrSNN-CP-NG) have been proven to obtain better predictive performance than the direct version of TrSNN without changes of the NeuCube structure.

The next chapter will demonstrate an empirical study on transfer learning across multiple subjects to further investigate the feasibility of these transfer learning models.

5.6 Contribution

In this chapter, I have made the following contributions:

- Designed an empirical study for transfer learning across tasks in one subject
- Analysed the learning patterns in SNNcube captured during the learning process in SNN models

Chapter 6 Modelling Transfer Learning Across Multiple Subjects

6.1 Introduction

In this chapter, the proposed transfer learning methods are further examined on the second case study, transfer learning across multiple subjects, and the hypothesis is that the proposed transfer learning approaches can be also successfully used in the subject-to-subject transfer learning scenario. In this study, I constructed optimal SNN models and trained them with data from multiple subjects. The models are also evaluated in a three-phase analysis, including SNNcube patterns in an unsupervised mode, output layer patterns in a supervised mode, and the final experiment results to evaluate the transfer learning performance.

6.2 Experimental Design

In this case study, the same EEG data that has been used in Chapter 5 is selected here again, and the description of EEG dataset and preprocessing steps were presented in Chapter 5. To validate the efficacy of the proposed approaches in transfer learning across multiple subjects, the SNNcube and deSNN classifier were first trained with samples from source subjects. After that, samples from the other three target subjects were presented one-by-one to the model without any data reinforcement from previous subjects with the goal of transferring some knowledge acquired from the source subject to the target subjects. A series of experiments were

conducted to validate the performance of the proposed transfer learning approaches across four selected subjects for two classes, class 2 and class 3 using the following schemes:

- **Baseline**: batch learning of all subjects' data using NeuCube
- **lmSNN**: Incremental learning of each subject's data sequentially in deSNN classifier only
- **TrSNN**: Transfer learning of each subject's data sequentially in unsupervised mode and train deSNN classifier
- **TrSNN-CP**: Transfer learning of each subject's data sequentially in unsupervised mode and deSNN classifier, along with weight connection pruning in the SNNcube after learning of each task
- **TrSNN-CP-NG**: Transfer learning of each subject's data sequentially in unsupervised mode and deSNN classifier, along with weight connection pruning in the SNNcube and neuron aggregation in the deSNN classifier
- **TrSNN-OP**: Transfer learning of each subject's data sequentially in unsupervised mode and deSNN classifier, along with connection weight pruning between SNNcube and the output neurons in the classifier
- **TrSNN-OP-NG**: Transfer learning of each subject's data sequentially in unsupervised mode and deSNN classifier, along with connection weights pruning between SNNcube and output neurons and neuron aggregation when training deSNN classifier

The flowchart of the preparation of training and testing datasets for different schemes is presented in Figure 6-1. The encoding procedure to transform input

signals into spike trains was a threshold-based encoding algorithm as explained in Chapter 3 and a fixed threshold was performed on each incoming subject.

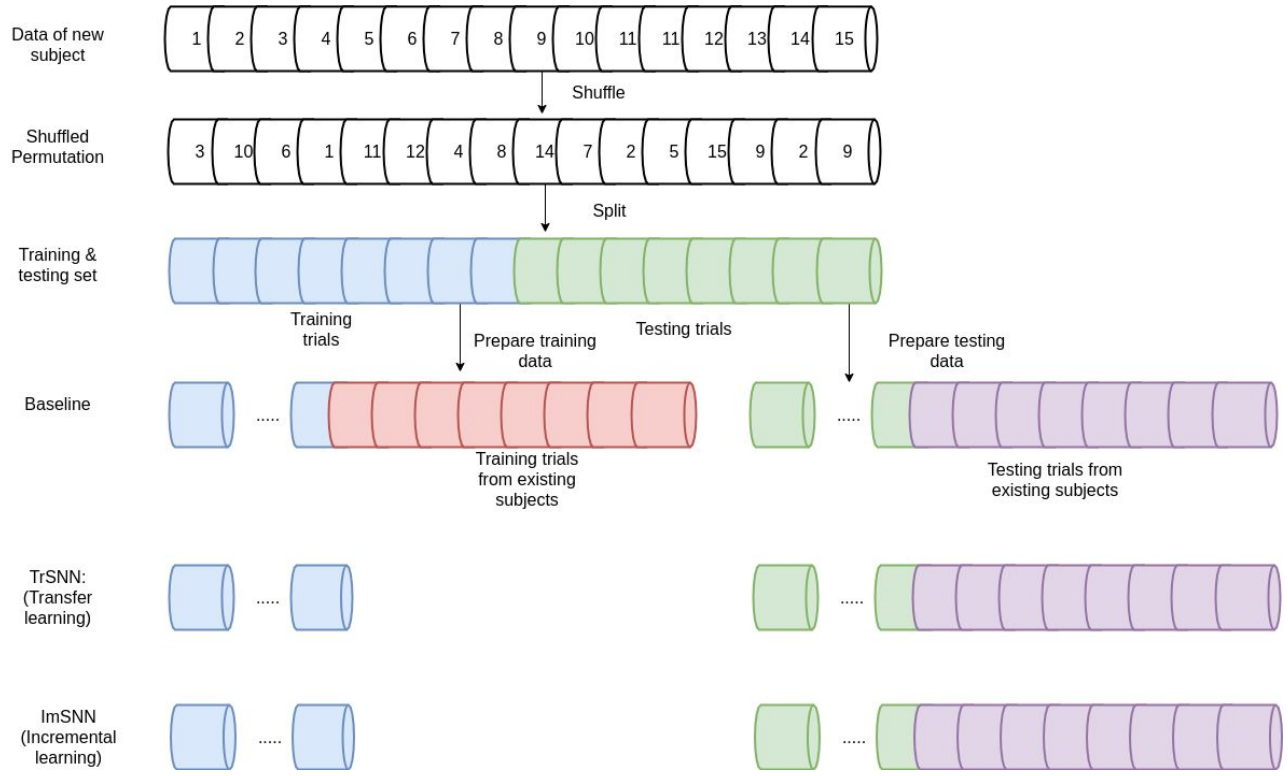

Figure 6-1 The flowchart of preparation of training and testing datasets for subject-to-subject transfer learning scenarios.

To demonstrate the performance of different schemes, both transfer learning model and baseline model were performed using the same initial connection weights. The NeuCube models were saved at each subject change, and the testing results of all previously seen subjects were evaluated. All experiments were performed five times and reported the test accuracy and its standard deviation on all tasks.

The parameter optimization phase is carried out on TrSNN models only. Parameters Drift, Mod, pruning percentage p, and pruning threshold θ were chosen using grid

search and picked the value of the best 3 fold cross-validation accuracy across all hyperparameters. For the similarity parameter (SIM) in the neuronal aggregation procedure of the deSNN classifier, a slightly different optimization scheme was used. The models were first trained over a range of similarity parameters along with the hyperparameters optimized as above using grid search. The optimal similarity parameter (SIM) was chosen as the value of the best training accuracy to make sure the performance of the neuron aggregation scheme performs on par with the model trained without neuron aggregation on the training dataset. Furthermore, baseline and ImSNN models were also conducted with the default parameters for comparison. The details of the default parameters are presented in Appendix B, Section B-1. For TrSNN experiments, there are some additional optimized parameters, which is summarized in Table 6-1.

Table 6-1 Parameter settings for each TrSNN experiment.

	Parameters
TrSNN-CP	Drift: 0.005 Mod: 0.8 SNNcube pruning percentage: 0.7
TrSNN-CP-NG	Drift: 0.005 Mod: 0.8 SNNcube pruning percentage: 0.7 SIM parameter: 2.5
TrSNN-OP-NG	Drift: 0.005 Mod: 0.8 Output layer pruning threshold: 0.533
TrSNN-OP-NG	Drift: 0.005 Mod: 0.8 Output layer pruning threshold: 0.533 SIM parameter: 2.5

6.3 Experimental Results

The experimental results was organised in a two-phase analysis as follows:

- Analysis of SNNcube patterns in an unsupervised mode
- Analysis of output layer patterns in a supervised mode
- Analysis of the results on test data to evaluate the transfer learning performance

6.3.1 Analysis of SNNcube Patterns in an Unsupervised Mode

The weights connectivity of SNNcube after connection weights pruning at each stage of the subject learning process for two classes, class 2 and class 3 are visualized in Figure 6-2 and Figure 6-3, respectively. It can be seen from Figure 6-2 and Figure 6-3 (a)-(d) that stronger brain connectivity is observed with further trained SNNcubes. For class 3 (Figure 6-3 (a)-(d)), the connections were particularly enhanced between neurons located in the areas of Occipital and Posterior, which were less observed in the case of class 2 (Figure 6-2 (a)-(d)).

To perform a better analysis of changes in SNNcube between subjects, the differences between the SNNcube for each stage of the learning process were computed through subtracting with the previous trained cube, which allows visualising the changes in neural connectivity as a result of transfer learning over time. In the graphs shown in Figure 6-2 (e), when the SNNcube was trained on subject 10, greater connections were mostly observed around the neurons positioned in Occipital and Temporal areas compared to other areas. As shown in

Figure 6-2 (f) for the subject 11, the neuron connectivity was enhanced around the Parietal, Sublobar, Temporal, Posterior and Occipital regions. Figure 6-2 (g) illustrates that EEG channels positioned in the Talairach areas associating with Frontal and Posterior lobes represented the most differences between subject 11 and subject 12. The connection weight varied to different degrees as new subjects were added, demonstrating the difference in activity of brain areas across subjects throughout the entire training process. Similarly, the brain states when executing task 3 were compared in Figure 6-3 (e-g). Further trained SNNcube for subject 11 resulted in a similar pattern of changes in some regional activation compared with class 2, while significant connectivity changes in some brain regions were observed for subject 10 and subject 12. Further analysis of the connectivity patterns of SNNcube can be performed in several ways, such as quantitative analysis. Another more efficient way to analyse the learning patterns is through deep knowledge representation, which will be discussed in Chapter 7.

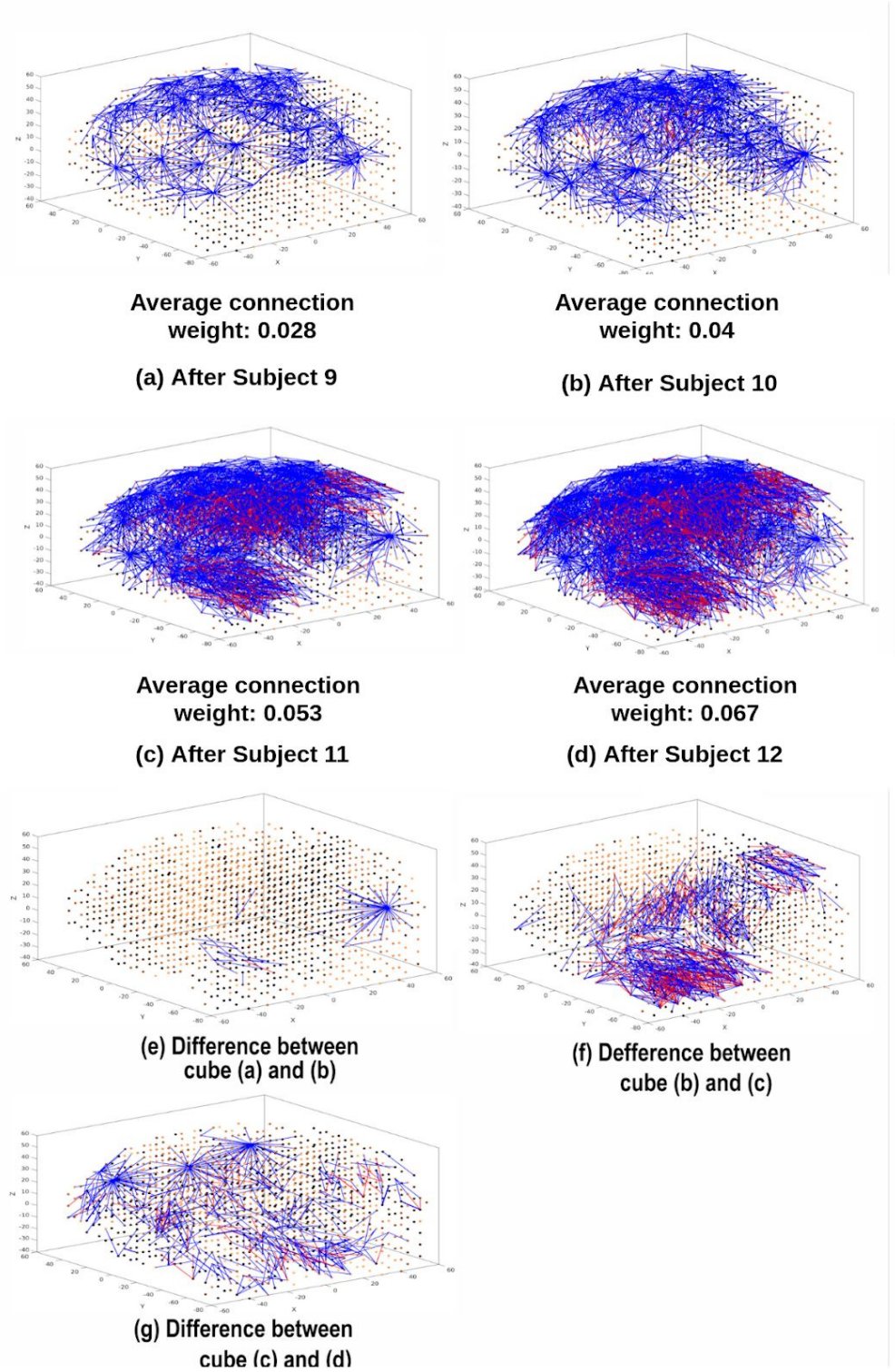

Figure 6-2 The connection weights of the pruned SNNcube for class 2 in TrSNN models trained (a) after subject 9 (threshold: 0.3), (b) subject 10 (threshold: 0.4), (c) subject 11 (threshold: 0.5), (d) after subject 12 (threshold: 0.6). Differences between the connectivity in the trained TrSNN models (e)(f)(g) (threshold: 0.3). The larger the number of new connections, the larger the difference between the new subject and old ones for the same task.

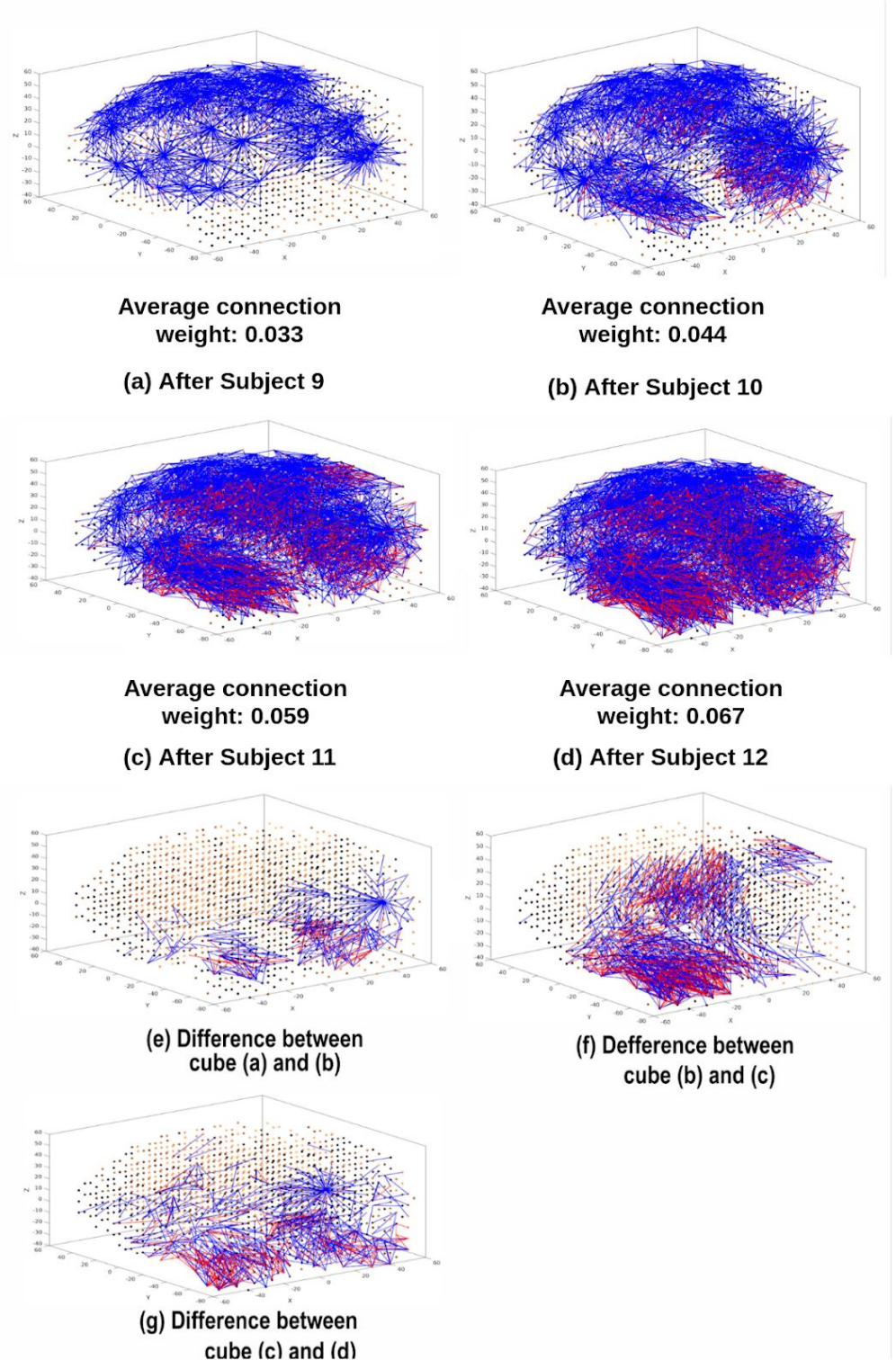

Figure 6-3 The connection weights of the pruned SNNcube for class 3 in TrSNN models trained (a) after subject 9 (threshold: 0.3), (b) subject 10 (threshold: 0.4), (c) subject 11 (threshold: 0.5), (d) after subject 12 (threshold: 0.6). Differences between the connectivity in the trained TrSNN models (e)(f)(g) (threshold: 0.3). For both task 2 (Figure 6-2) and task 3 (this figure) the larger differences are observed when Subject 11 data is learned in the SNNcube.

6.3.2 Analysis of Output Layer Patterns in a Supervised Mode

As mentioned in Chapter 3, all neurons in the SNNcube are connected to every neuron in the output layer. Therefore, after output layer pruning, inactive neurons in output layer, which did not have activate connection with SNNcube, were pruned, which reduced the output connections for all samples from 137970 to 108727, as shown in Table 6-2, a significant reduction of the dimensionality of the space for classification.

Table 6-2 The number of connections of the output layer for the TrSNN model before and after pruning.

Output Layer Pruning		
Model	**Total**	**Density**
Complete	137970	100%
Pruned	108727	79%

The number of output neurons before and after aggregation is illustrated in Appendix B, Figures B-2.

6.3.3 Analysis of the Results on Test Data to Evaluate the Transfer Learning Performance

The performance at the end of the training process was first evaluated using the overall accuracy and F-score achieved by each scheme across all four subjects, as shown in Figure 6-4 and Figure 6-5. TrSNN-CP resulted in the second highest classification performance by achieving up to 88.89% accuracy, suggesting that TrSNN-CP achieved better performance compared with other transfer learning or

incremental learning schemes. In terms of the performance for each subject, subject 11 has a lower performance than other subjects. This can be explained by the large interclass ambiguity between the class 2 and 3 for subject 11 as shown in Figure 6-2 (f) and Figure 6-3 (f). These two classes exhibited high similarity with the areas of neurons in the SNNcube for the activation of this output neuron, hence the misclassification was mainly among the subject 11. Figure 6-4 and Figure 6-5 showed that the TrSNN-CP and TrSNN-CP-NG outperformed the incremental learning method (ImSNN) with an average accuracy improvement of 4%, which is opposite from the experiments in the previous chapter.

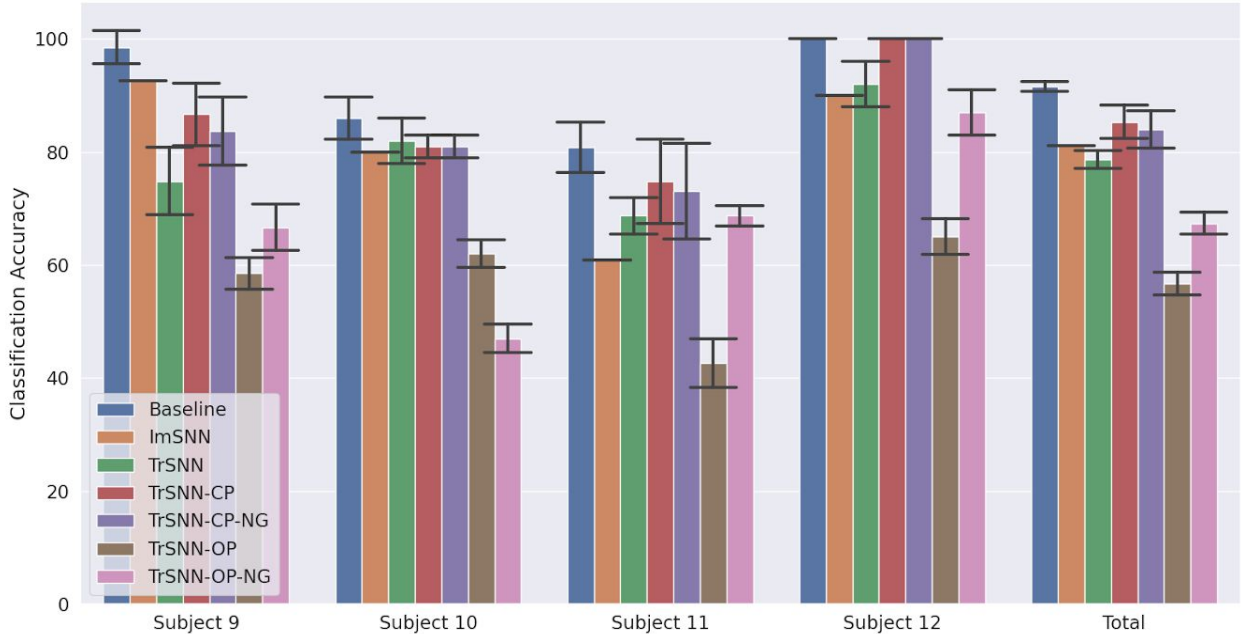

Figure 6-4 Final average accuracy for each experiment after training four subjects using two classes (class 2 and class 3).

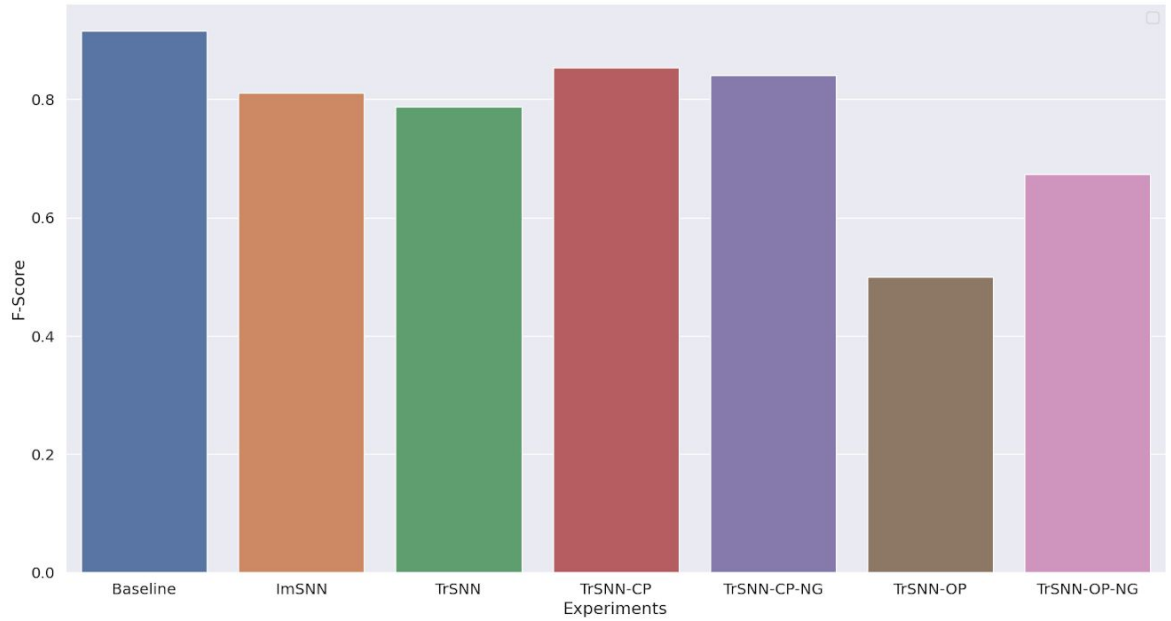

Figure 6-5 Final F-score for each experiment after training four subjects using two classes (class 2 and class 3).

In order to further analyze the accuracy of per class accuracy across subjects, per class accuracy of three schemes at the end of the training process was evaluated across all four subjects. It can be seen from Figure 6-6 that there was no significant catastrophic forgetting when the new subjects are learned using the TrSNN model. This might suggest that the feature of new subjects has no significant difference from previously observed examples. Unexpectedly, there were sharp drops in accuracy of class 3 for subject 11 when trained with TrSNN mode. The TrSNN-CP model performed on par with the baseline model for all subjects under most circumstances. Subject 11 in particular had a large increase in accuracy for class 3 when the SNNcube pruning was applied, while retaining the accuracy of other subjects.

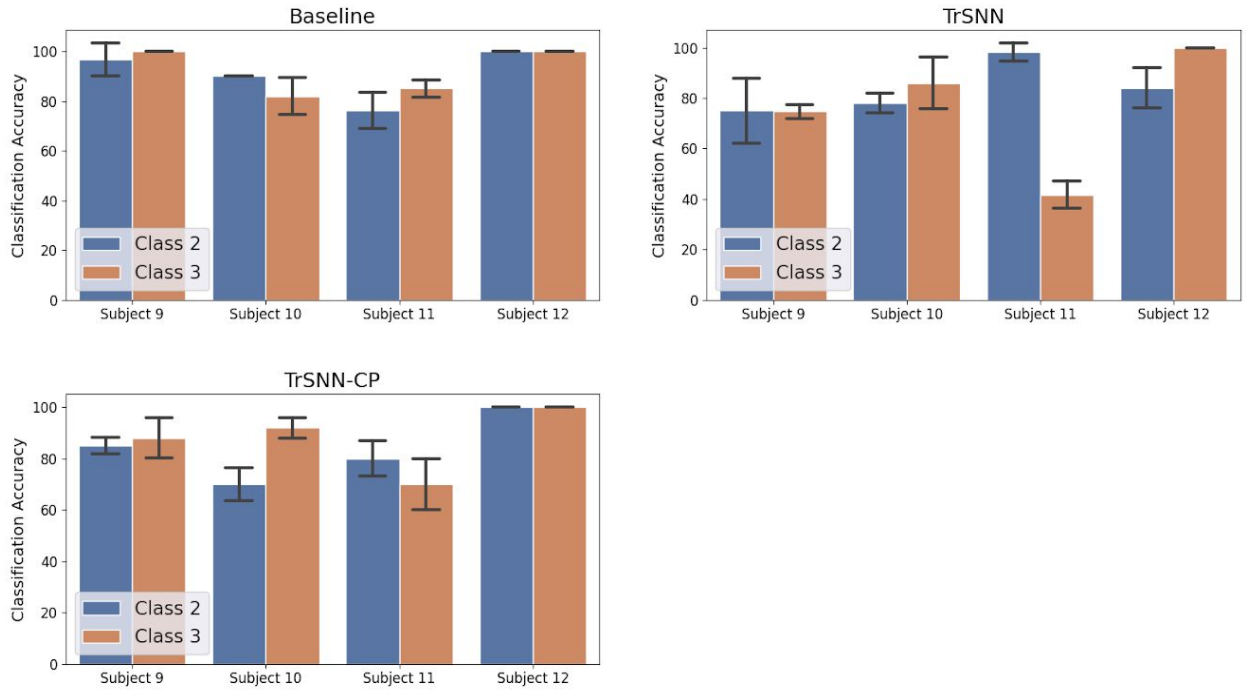

Figure 6-6. Per class accuracy for each subject trained with baseline, TrSNN, and TrSNN-CP models.

6.4 Chapter Summary

In this chapter, I have shown that the proposed TrSNN model with SNNcube pruning has a better adaptability to the novel information received from the new subjects by cutting off certain connections in the network, while preserving useful knowledge from previously learned subjects, allowing rapid learning from new information. Now the question is: how the SNN models can be further investigated for a better understanding of learning patterns that lead to a successful transfer learning and which parts of the brain area are responsible for that? In the next chapter, a deep spatio temporal rules extraction algorithm will be introduced. This allows for knowledge representation in the models and contributes to the interpretation of knowledge being transferred when training new tasks or subjects subsequently.

6.5 Contribution

In this chapter, I have made the following contributions:

- Designed an empirical study on transfer learning across multiple subjects
- Analysed the learning patterns in SNNcube captured during the learning process in SNN models

Chapter 7 Methods and Algorithms for Knowledge Representation in BI-SNN as Spatial Temporal Rules (STR)

7.1 Introduction

In the previous two chapters, task to task and subject to subject experiments were conducted to examine the proposed transfer learning approaches. Besides the model accuracy, a better interpretation of the model is also essential. In this chapter, I proposed a deep knowledge extraction and representation method using BI-SNN architectures. This knowledge representation method is a technique to extract spatial temporal rules from deep knowledge, improving the level of interpretability of learning patterns that lead to a successful transfer learning and which parts of the brain area are responsible for that. In addition, this algorithm was applied to perform further analysis for the evolving patterns in transfer learning models, which has not been interpreted in depth in the previous two experiments presented in Chapter 5 and Chapter 6.

7.2 Deep Spatio Temporal Rules Extraction Approach

The concept of deep knowledge representation in SNN is first introduced in (Kasabov, 2019). Deep knowledge extraction and representation in a trained SNN model is algorithmically described in Algorithm 6, in which activation activities of

different clusters of brain area at slightly different times are extracted numerically, and use it to represent spatiotemporal relationships during a cognitive task. In BI-SNN framework, each spiking neuron was annotated spatially by its corresponding anatomical region in the human brain using a 3D coordinate in the Talairach space (Talairach & Tournoux, 1988). The spatial mapping of EEG data is presented in Appendix A. After unsupervised learning in SNNcube with SNNcube pruning, the firing rate in each time bin was calculated along with neuron aggregation during the supervised learning stage. The neuron aggregation was employed since merged output neurons represent clusters of prototypes in a transformed space, and this transformed space can be used to discover deep knowledge. More specifically, for each aggregated output neuron, the average firing rate of the spatial cluster in each time bin can be calculated based on the number of times a pre-synaptic neurons spike. The spatial clusters were considered as active if the normalized firing rate surpasses the rule threshold. After the spatio-temporal analysis, deep knowledge is formed through the information obtained above.

The activation of different clusters of neurons associated with each output class sample (prototype) at slightly different time bins can be analysed and deep spatio-temporal rules can be extracted. For each output class sample S, a chain of activated clusters of brain areas was formed over each time frame. A fuzzy rule-base associated with this output class sample can be extracted and presented in the following form (Kasabov, 2019):

IF (the firing rate of $area_1$ is A and $area_2$ is B and $area_3$ is F, at time about T1)

AND (the firing rate of $area_1$ is A and $area_2$ is B and $area_3$ is F, at time about T2)

AND (the firing rate of $area_1$ is A and $area_2$ is B and $area_3$ is F, at time about T3)

AND (the firing rate of $area_1$ is A and $area_2$ is B and $area_3$ is F, at time about T4)

THEN (The output class prototype is class C)

Where A, B and F are fuzzy values represented by their membership functions, C is its corresponding class label, and $area_i$ indicates the set of brain areas that are activated at time T. For instance, $area_i$ at the lobe level of the hierarchy can be represented using the following subset of brain areas:

$$area_i \subset \{Anterior\ Lobe,\ Frontal\ Lobe,\ Frontal-Temporal\ Space,\ Limbic\ Lobe,\ Medulla,\ Midbrain,\ Occipital\ Lobe,\ Parietal\ Lobe,\ Pons,\ Posterior\ Lobe,\ Sub-lobar,\ Temporal\ Lobe\}$$

An example of fuzzy membership functions is shown in Figure 7-1.

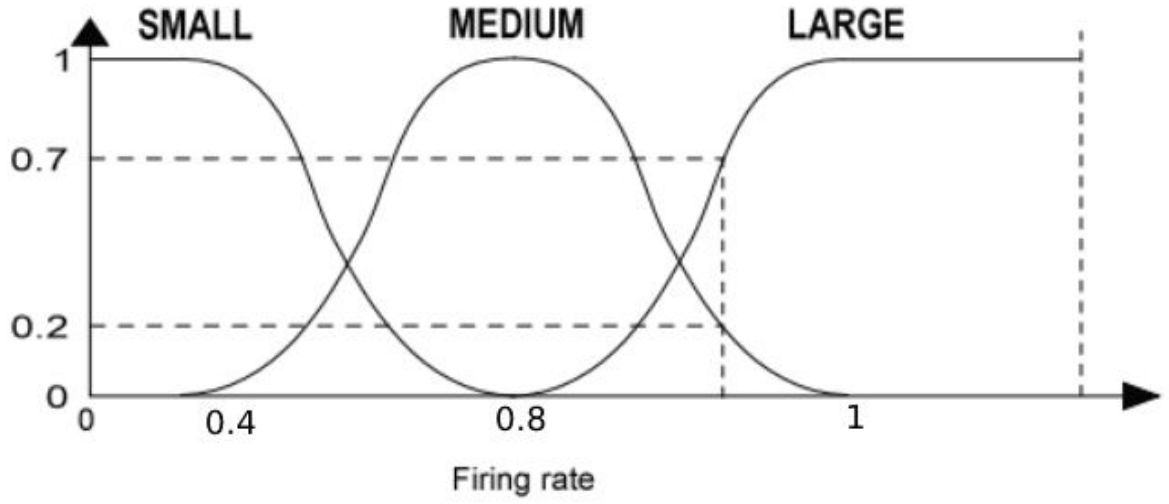

Figure 7-1 Example of fuzzy gaussian membership functions that represent a variable firing rate.

In the next two sections, this spatio temporal rules extraction algorithm was applied to further analyse both task-to-task and subject-to-subject experiments. The temporal resolution for this analysis was four time bins (t = {0.5s, 1s, 1.5s, 2s}), and the spatial resolution was the lobe level in the Talairach Brain Atlas (Talairach & Tournoux, 1988). The rule threshold th_{rule} that used to select spatial clusters for spatial temporal rule generation was fixed to 0.3.

Algorithm 6 Algorithm for Extracting Deep Spatiotemporal Rules

Input: input signals; 3-D coordinates of the neurons in SNN; spatial cluster ids of each neuron in SNN; 3-D coordinates of each input channel

Parameters: th_{rule}, threshold to select spatial clusters to generate spatiotemporal rule

Output: a set of deep spatiotemporal rules

Generic NeuCube process:

1: Encode input signals into spikes
2: Initialise SNN using 3-D coordinates of spiking neurons
3: Map input channel locations into 3-D space of SNN
4: Perform unsupervised learning in SNN using STDP learning rule with cube pruning
5: Perform deSNN supervised learning with neuron aggregation

Extraction of deep spatio temporal rules:

1: **for** every output neurons in the output layer **do**
2: Get connection weights between neurons of SNNcube and output neuron w
3: Get average firing rate of each spatial cluster in each time bin $a_{n,t} = \sum_{j=1}^{k} \frac{r_{n,j}}{k}$, where $r_{n,j}$ is the number of times a pre-synaptic neuron in the n^{th} spatial cluster spike during the time bin t; k is the number of neurons in the n^{th} spatial cluster
4: Normalize the average firing rate of each spatial cluster in each time bin $q_{n,t} = \frac{a_{n,t} - a_{min}}{a_{max} - a_{min}}$
5: Select spatial clusters using $n = f(q_{n,t} : \forall n)$ where t is the event time bin, $n \in E_t$ and $f(x) = \begin{cases} n, & \text{if } x \geq th_{rule}. \\ \{\}, & \text{otherwise.} \end{cases}$
6: **end for**
7: Generate a set of fuzzy/crisp rules

Table 7-1 The proposed deep spatial temporal rule extraction algorithm.

7.3 Spatial Temporal Rules for Task To Task Transfer in One Subject

7.3.1 Functional Organisation of Neural Clusters

For each output class sample, a subset of brain areas that indicate a specific combination of activation level during the execution of a cognitive task were selected. In order to perform a better quantitative analysis of the organization of neural clusters across tasks, the average firing rates of different spatial clusters of brain areas in each time bin are first calculated, and the difference of firing rate for each stage of the learning process were computed through subtracting with the firing rate for previous trained model, as shown in Figure 7-2. Different activation level of slightly different clusters of neurons at different times are observed, for instance, strong firing rates were created around the Frontal and Limbic lobes in Figure 7-2 (b) while Frontal-temporal lobe was more active in Figure 7-2 (a), indicating different knowledge were transferred at different stage of the training process.

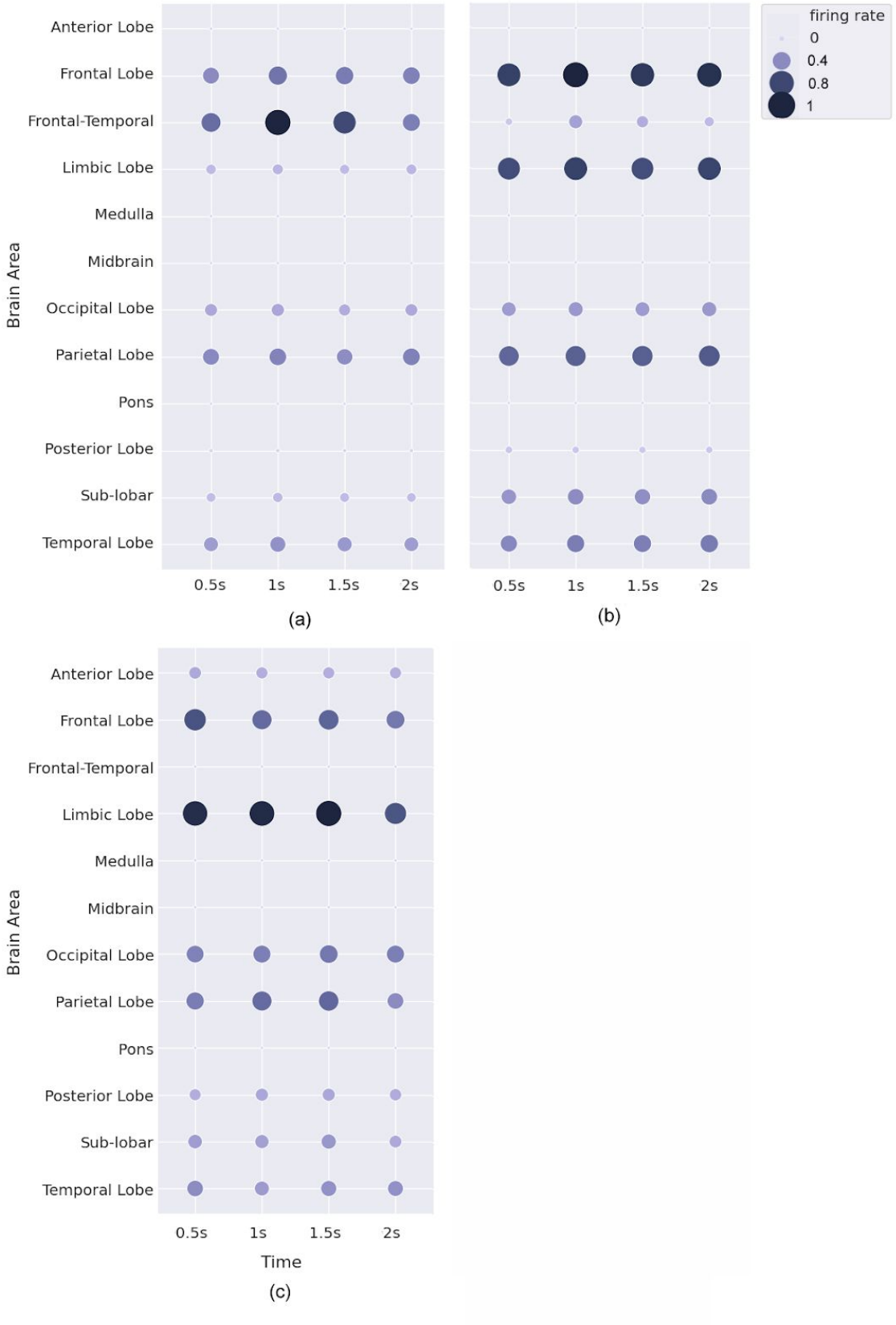

Figure 7-2 Difference in firing rate of brain areas between the SNN models trained (a) after class 1 and after class 2 (b) after class 2 and after class 3, (c) after class 3 and after class 4.

7.3.2 Extraction of Fuzzy Rules

The knowledge obtained above can further be converted into meaningful symbolic representation, in which a fuzzy rule can be formed by using the activation of slightly different clusters of neurons at slightly different times. Below are the fuzzy rules obtained from Figure 7-2 (a) and Figure 7-2 (b).

A fuzzy rule for sample in Figure 7-2 (a) can be written as,

- $area_1(T1)$ = *{Temporal Lobe}*

  $area_2(T1)$ = *{Frontal-Temporal Space, Frontal Lobe, Posterior Lobe}*
- $area_1(T2)$ = *{Temporal Lobe}*

  $area_2(T2)$ = *{ Frontal Lobe, Temporal Lobe}*

  $area_3(T2)$ = *{Frontal-Temporal Space}*
- $area_1(T3)$ = *{Parietal Lobe, Temporal Lobe}*

  $area_2(T3)$ = *{Frontal-Temporal Space, Frontal Lobe}*
- $area_1(T4)$ = *{Temporal Lobe}*

  $area_2(T4)$ = *{Frontal-Temporal Space, Frontal Lobe, Posterior Lobe}*

IF (the firing rate of $area_1(T1)$ is SMALL and $area_2(T1)$ is MEDIUM, at time about 0.5s)

AND (the firing rate of $area_1(T2)$ is SMALL and $area_2(T2)$ is MEDIUM and $area_3(T2)$ is HIGH, at time about 1s)

AND (the firing rate of $area_1(T3)$ is SMALL and $area_2(T3)$ is MEDIUM, at time about 1.5s)

AND (the firing rate of $area_1(T4)$ is SMALL and $area_2(T4)$ is MEDIUM, at time about 2s)

THEN (This is the knowledge that transfers from the model trained with class 1 to the model that continues to train class 2 sequentially) .

A fuzzy rule for sample in Figure 7-2 (b) can be written as,

- $area_1(T1)$ = *{Occipital Lobe, Sub-lobar}*

  $area_2(T1)$ = *{Temporal Lobe, Limbic Lobe, Parietal Lobe}*

  $area_3(T1)$ = *{Frontal Lobe}*
- $area_1(T2)$ = *{Occipital Lobe, Sub-lobar}*

  $area_2(T2)$ = *{Temporal Lobe, Parietal Lobe}*

  $area_3(T2)$ = *{Frontal Lobe, Limbic Lobe}*
- $area_1(T3)$ = *{Occipital Lobe, Sub-lobar}*

  $area_2(T3)$ = *{Temporal Lobe, Limbic Lobe, Parietal Lobe}*

  $area_3(T3)$ = *{Frontal Lobe}*
- $area_1(T4)$ = *{Occipital Lobe, Sub-lobar}*

  $area_2(T4)$ = *{Temporal Lobe, Parietal Lobe}*

  $area_3(T4)$ = *{Frontal Lobe, Temporal Lobe}*

IF (the firing rate of $area_1(T1)$ is SMALL and $area_2(T1)$ is MEDIUM and $area_3(T1)$ is HIGH, at time about 0.5s)

AND (the firing rate of $area_1(T2)$ is SMALL and $area_2(T2)$ is MEDIUM and $area_3(T2)$ is HIGH, at time about 1s)

AND (the firing rate of $area_1(T3)$ is SMALL and $area_2(T3)$ is MEDIUM and $area_3(T3)$ is HIGH, at time about 1.5s)

AND (the firing rate of $area_1(T4)$ is SMALL and $area_2(T4)$ is MEDIUM and $area_3(T4)$ is HIGH, at time about 2s)

THEN (This is the knowledge that transfers from the model trained with class 1 and class 2 to the model that continues to train class 3 sequentially)

7.4 Spatial Temporal Rules for Subject To Subject Transfer

7.4.1 Functional Organisation of Neural Clusters

In order to perform a better quantitative analysis of the organization of neural clusters between subjects, the average firing rates of different spatial clusters of brain areas in each time bin for class 2 and class 3 are first calculated and the difference of firing rate for each stage of the learning process were computed through subtracting with the firing rate for previous trained model and visualized in Figure 7-3 and Figure 7-4 respectively. In the graphs shown in Figure 7-3 (a), the common knowledge for the model trained with subject 9 and model that continue to trained with subject 10 was positioned around the areas in the Talairach areas associating with Anterior Lobe, Medulla, Midbrain, Pons, Posterior Lobe at each time bin. In Figure 7-3 (b), strong firing rates were mostly created around the Medulla, Posterior, Pons and Occipital regions. Figure 7-3 (c) depicts that those neurons with a high firing rate positioned around the Sub-lobar, Pons and Temporal Lobe after 1s, Anterior Lobe in 1s and 2s, Limbic Lobe in 2s, while neural clusters in Occipital and

Posterior Lobe are activated at all time bins. Similarly, different activation activities of different clusters of brain area at slightly different times when executing task 3 were observed in Figure 7-4 (a-c).

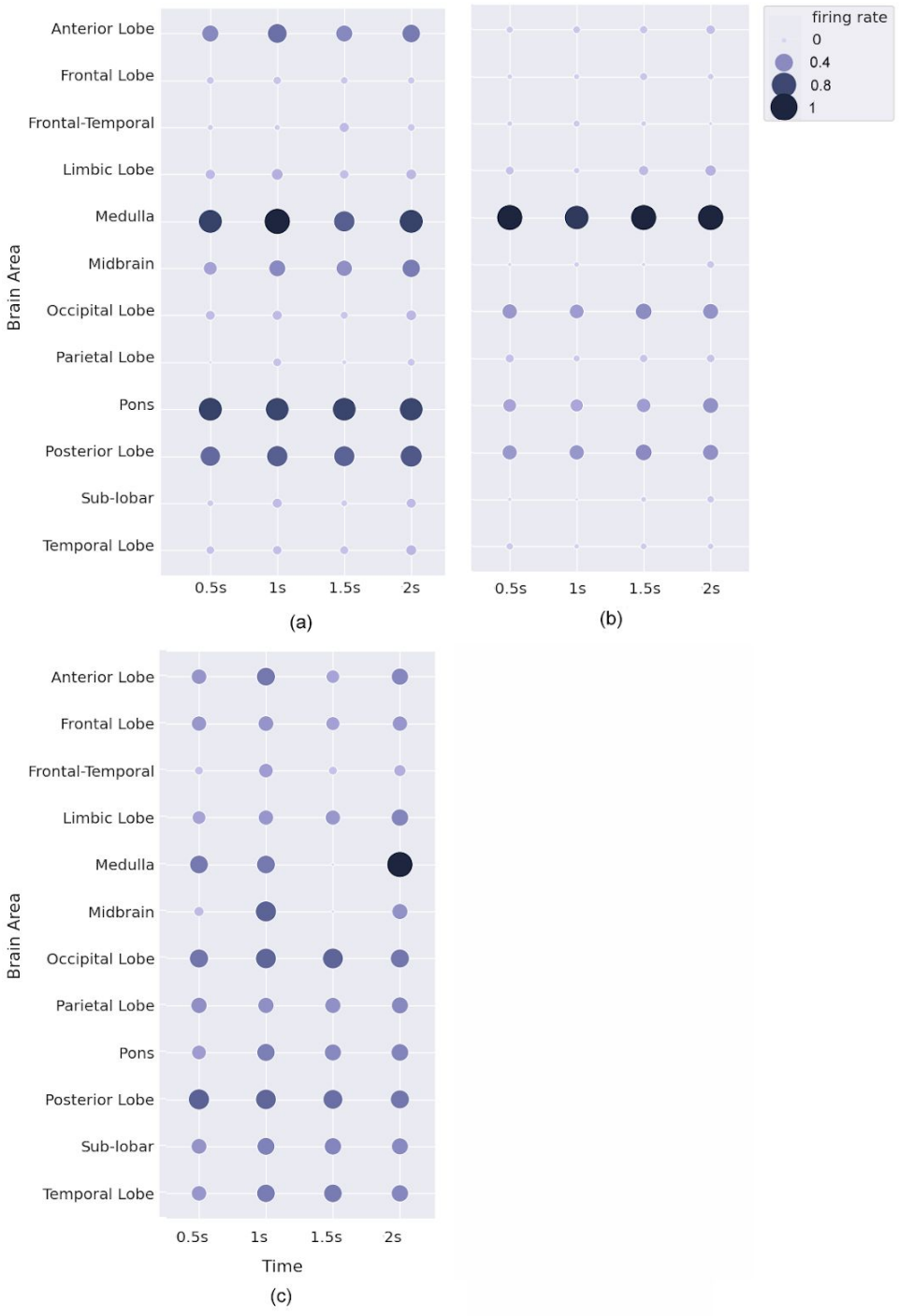

Figure 7-3 Difference in firing rate of brain areas when executing class 2 between the trained SNN models (a) after subject 9 and after subject 10 (b) after subject 10 and after subject 11, (c) after subject 11 and after subject 12.

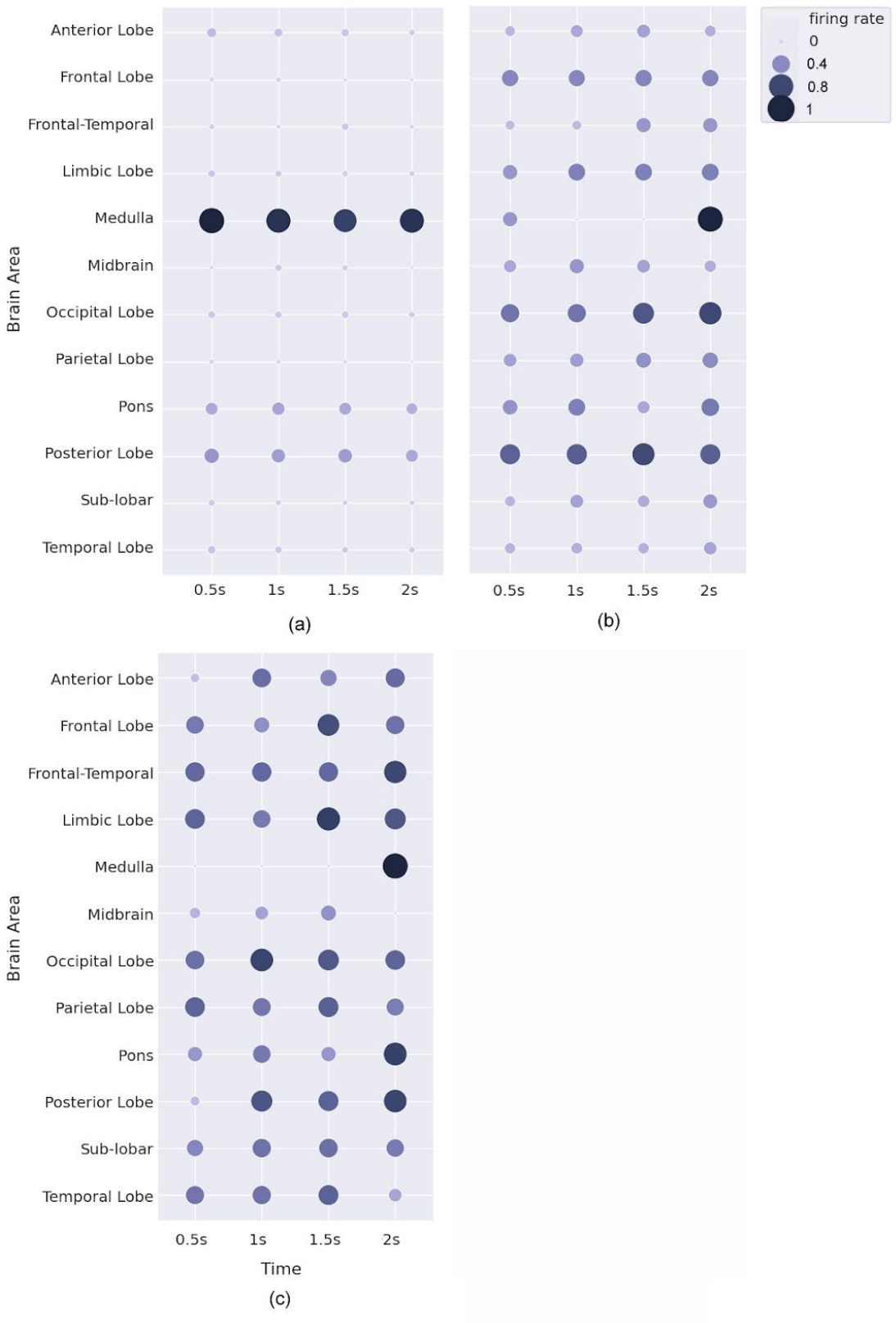

Figure 7-4 Difference in firing rate of brain areas when executing class 3 between the trained SNN models (a) after subject 9 and after subject 10 (b) after subject 10 and after subject 11, (c) after subject 11 and after subject 12.

7.4.2 Extraction of Fuzzy Rules

The knowledge obtained above can further be converted into meaningful symbolic representation, in which a fuzzy rule can be formed by using the activation of slightly different clusters of neurons at slightly different times. Below are the fuzzy rules obtained from Figure 7-3 (c) and Figure 7-4 (c).

A fuzzy rule for sample in Figure 7-3 (c) can be written as,

- $area_1(T1)$ = *{Frontal Lobe, Anterior Lobe, Temporal Lobe, Parietal Lobe, Pons, Sub-lobar}*

  $area_2(T1)$ = *{Posterior Lobe, Occipital Lobe, Medulla}*
- $area_1(T2)$ = *{Limbic Lobe, Frontal Lobe, Frontal-Temporal Space, Parietal Lobe}*

  $area_2(T2)$ = *{Temporal Lobe, Occipital Lobe, Medulla, Sub-lobar, Midbrain, Posterior Lobe, Pons, Anterior Lobe}*
- $area_1(T3)$ = *{Parietal Lobe, Limbic Lobe}*

  $area_2(T3)$ = *{Temporal Lobe, Pons, Occipital Lobe, Sub-lobar, Posterior Lobe}*
- $area_1(T4)$ = *{Midbrain, Frontal Lobe}*

  $area_2(T4)$ = *{Anterior Lobe, Posterior Lobe, Limbic Lobe, Occipital Lobe, Pons, Parietal Lobe, Temporal Lobe, Sub-lobar}*

  $area_3(T4)$ = *{Medulla}*

IF (the firing rate of $area_1(T1)$ is SMALL, $area_2(T1)$ is MEDIUM, at time about 0.5s)
AND (the firing rate of $area_1(T2)$ is SMALL, $area_2(T2)$ is MEDIUM, at time about 1s)
AND (the firing rate of $area_1(T3)$ is SMALL, $area_2(T3)$ is MEDIUM, at time about 1.5s)

AND (the firing rate of $area_1(T3)$ is SMALL, $area_2(T3)$ is MEDIUM, $area_2(T3)$ is HIGH, at time about 2s)

THEN (This is the knowledge for class 2 that transfers from the model trained with subject 9, 10, and 11 to the model that continues to train subject 12 sequentially) .

A fuzzy rule for sample in Figure 7-4 (c) can be written as,

- $area_1(T1)$ = *{Pons}*

  $area_2(T1)$ = *{Frontal Lobe, Temporal Lobe, Frontal-Temporal Space, Limbic Lobe, Occipital Lobe, Parietal Lobe, Sub-lobar}*
- $area_1(T2)$ = *{Frontal Lobe}*

  $area_2(T2)$ = *{Temporal Lobe, Limbic Lobe, Sub-lobar, Posterior Lobe, Frontal-Temporal Space, Pons, Parietal Lobe, Anterior Lobe}*

  $area_3(T2)$ = *{Occipital Lobe}*
- $area_1(T3)$ = *{Pons, Midbrain}*

  $area_2(T3)$ = *{Frontal Lobe, Temporal Lobe, Anterior Lobe, Occipital Lobe, Parietal Lobe, Frontal-Temporal Space, Sub-lobar, Posterior Lobe}*

  $area_3(T3)$ = *{Limbic Lobe}*
- $area_1(T4)$ = *{Anterior Lobe, Limbic Lobe, Occipital Lobe, Frontal Lobe, Parietal Lobe, Sub-lobar}*

  $area_2(T4)$ = *{Posterior Lobe, Frontal-Temporal Space, Pons, Medulla}*

IF (the firing rate of $area_1(T1)$ is SMALL, at time about 0.5s)

AND (the firing rate of $area_1(T2)$ is SMALL and $area_2(T2)$ is MEDIUM and $area_3(T2)$ is HIGH, at time about 1s)

AND (the firing rate of $area_1(T3)$ is SMALL and $area_2(T3)$ is MEDIUM and $area_3(T3)$ is HIGH, at time about 1.5s)

AND (the firing rate of $area_1(T4)$ is MEDIUM and areas $area_2(T4)$ is LARGE, at time about 2s)

THEN (This is the knowledge for class 3 that transfers from the model trained with subject 9, 10, and 11 to the model that continues to train subject 12 sequentially)

7.5 Chapter Summary

In this chapter, I proposed a new deep spatio temporal rules extraction approach based on the SNN architecture. This resulted in a better interpretation of SNN learning patterns. It also contributes to knowledge representation in SNN architectures, allowing further analysis on how the trained transfer learning models exchange information and transfer knowledge between tasks or subjects.

7.6 Contribution

In this chapter, I have made the following contributions:

- Proposal of a new deep spatio temporal rules extraction approach based on the SNN architecture
- Designed a study on the proposed spatio temporal rules extraction approach based on the task to task and subject to subject experiments

Chapter 8 Conclusions and Recommendations for Future Work

8.1 Introduction

This chapter discusses the key findings and contributions of this thesis. The main limitations of this work are then discussed along with an overview of future implications.

8.2 Aim and Methodological Approach

This thesis aims to adapt the NeuCube Brain-Inspired Spiking Neural Network (BI-SNN) architecture for transfer learning schemes. I proposed a family of transfer learning methods based on the NeuCube framework that resulted in a better adaptability to the novel information. The proposed approaches were experimentally validated using the upper limb movement dataset on two empirical studies, including transfer learning across tasks in one subject and transfer learning across subjects in one task. Additionally, I proposed a new spatial-temporal rule based on SNN architecture that offers an improved level of interpretability about how the transfer learning models exchange information.

8.3 Empirical and Theoretical Contributions

The contributions of this thesis are:

1. Transfer learning in BI-SNN models

I adapted the NeuCube framework to be used on transfer learning environments, which resulted in a better adaptability to the novel information received from the new tasks or subjects, achieving up to 81.96% accuracy for task-to task transfer learning in one subject and 88.89% accuracy for subject-to-subject transfer learning case studies. The findings demonstrated that the new adaptations of SNN models are able to learn from new incoming tasks or subjects rapidly without requiring retraining from previous samples, while retaining the knowledge from previously learned tasks or subjects with less forgetting. Therefore, the achieved transfer learning in SNN models is a significant contribution for a further development of deep-learning in SNN architecture.

2. Knowledge representation in BI-SNN models

I proposed a new deep knowledge representation approach to extract spatial temporal rules from deep knowledge, enabling a better interpretation of learning patterns in the SNN models. This approach also contributed to further analysis on how the transfer learning models exchange information in order to trace the evolution of knowledge during transfer learning.

8.4 Limitation of the Thesis

The limitations of this thesis are:

1. Scope and Parameters of the Research

The temporal and spatial resolution of knowledge representation extracted from a trained SNN model was predefined manually. The optimization procedure needs to be further investigated to find the optimal space and time of knowledge. In addition, the data used in this thesis were only EEG data, however the proposed transfer learning methods need to be further evaluated to other types of spatio-temporal data, for instance, fMRI and gene data.

2. Methodological Point of View

A fixed threshold was used for encoding the input signals into spike trains for all tasks and subjects. However, the encoding procedure and its parameters should be adapted to different tasks or subjects. Assessment of different encoding methods and optimising their parameters is crucial for further development of transfer learning models.

8.5 Future Direction and Implications

There are some future directions that can be explored in the future as follows:

Bioinformatics: The proposed transfer learning methods presented in Chapter 5 can be further applied to the field of bioinformatics, for instance, a generic predictive system for early prediction of health risk factors. To this aim, I aim to investigate the feasibility of transfer learning using omics data, such as genomics or proteomics data.

Learning to learn in BI-SNN: For further development of the proposed methods presented in Chapter 5, I aim to enhance it towards learning to learn in SNN models. The SNN models are not only capable of transferring prior knowledge in order to learn new information more effectively but learning related tasks or subjects from very few examples.

References

Abbott, L. F. (1999). Lapicque's Introduction of the Integrate-and-Fire Model Neuron (1907). *Brain Research Bulletin*, *50*(5), 303-304.

Brett, M., Christoff, K., Cusack, R., & Lancaster, J. (2001). Using the Talairach Atlas with the MNI Template. *Neuroimage, 13*(6), 85-85.

Bruzzone, L., & Marconcini, M. (2009). Domain adaptation problems: A DASVM classification technique and a circular validation strategy. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 32*(5), 770-787.

Alamgir, M., Grosse–Wentrup, M., & Altun, Y. (2010, March). Multitask learning for brain-computer interfaces. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 17-24). JMLR Workshop and Conference Proceedings.

Azab, A. M., Toth, J., Mihaylova, L. S., & Arvaneh, M. (2018). A review on transfer learning approaches in brain–computer interface. *Signal Processing and Machine Learning for Brain-Machine Interfaces*, 81-98.

Allred, J. M., & Roy, K. (2020). Controlled Forgetting: Targeted Stimulation and Dopaminergic Plasticity Modulation for Unsupervised Lifelong Learning in Spiking Neural Networks. *Frontiers in neuroscience*, *14*.

Bohte, S. M., Kok, J. N., & La Poutré, J. A. (2000, April). SpikeProp: backpropagation for networks of spiking neurons. In *ESANN* (pp. 419-424).

Bohte, S. M. (2004). The Evidence for Neural Information Processing with Precise SpikeTimes: A Survey. *Natural Computing, 3*(2), 195-206.

Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J. M., Diesmann, M., Morrison, A., Goodman, P.H., Harris, F.C. & Zirpe, M. (2007). Simulation of networks of spiking neurons: a review of tools and strategies. *Journal of computational neuroscience, 23*(3), 349-398.

Bullmore, E., & Sporns, O. (2009). Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience, 10*(3), 186.

Bremner, A., Lewkowicz, D. & Spence, C. (2012). *Multisensory development*. Oxford University Press.

Cichon, J., & Gan, W. B. (2015). Branch-specific dendritic Ca 2+ spikes cause persistent synaptic plasticity. *Nature, 520*(7546), 180-185.

Cui, Y., Xu, Y., & Wu, D. (2019). EEG-based driver drowsiness estimation using feature weighted episodic training. *IEEE Transactions on Neural Systems and Rehabilitation Engineering, 27*(11), 2263-2273.

Capecci, E., Lobo, J. L., Laña, I., Espinosa-Ramos, J. I., & Kasabov, N. (2019). Modelling gene interaction networks from time-series gene expression data using evolving spiking neural networks. *Evolving Systems*, 1-15.

Delorme, A., & Makeig, S. (2004). EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. *Journal of neuroscience methods*, *134*(1), 9-21.

Delbruck, T., & Lichtsteiner, P. (2007). Fast Sensory Motor Control based on Event-Based Hybrid Neuromorphic-Procedural System. In *IEEE International Symposium on Circuits and Systems* (pp. 845-848). IEEE.

Davis, J., & Domingos, P. (2009, June). Deep transfer via second-order markov logic. In *Proceedings of the 26th annual international conference on machine learning* (pp. 217-224).

Dray, J., Capecci, E., & Kasabov, N. (2018, December). Spiking neural networks for cancer gene expression time series modelling and analysis. In *International Conference on Neural Information Processing* (pp. 625-634). Springer, Cham.

Doborjeh, Z. G., Kasabov, N., Doborjeh, M. G., & Sumich, A. (2018). Modelling peri-perceptual brain processes in a deep learning spiking neural network architecture. *Scientific reports, 8*(1), 1-13.

Fusi, S., Annunziato, M., Badoni, D., Salamon, A., & Amit, D. J. (2000). Spike-driven synaptic plasticity: theory, simulation, VLSI implementation. *Neural computation, 12*(10), 2227-2258.

Gerstner, W., & van Hemmen, J. L. (1992). Associative memory in a network of 'spiking'neurons. *Network: Computation in Neural Systems, 3*(2), 139-164.

Hebb, D. (1949). *The Organization of Behavior: A Neuropsychological Approach*. John Wiley & Sons.

Hu, J., Hou, Z. G., Chen, Y. X., Kasabov, N., & Scott, N. (2014, August). EEG-based classification of upper-limb ADL using SNN for active robotic rehabilitation. In *5th IEEE RAS/EMBS international conference on biomedical robotics and biomechatronics* (pp. 409-414). IEEE.

Hossain, I., Khosravi, A., & Nahavandhi, S. (2016, July). Active transfer learning and selective instance transfer with active learning for motor imagery based BCI. In *2016 International Joint Conference on Neural Networks (IJCNN)* (pp. 4048-4055). IEEE.

Hayes, T. L., Cahill, N. D., & Kanan, C. (2019, May). Memory efficient experience replay for streaming learning. In *2019 International Conference on Robotics and Automation (ICRA)* (pp. 9769-9776). IEEE.

He, H., & Wu, D. (2019). Transfer learning for Brain–Computer interfaces: A Euclidean space data alignment approach. *IEEE Transactions on Biomedical Engineering, 67*(2), 399-410.

He, H., & Wu, D. (2020). Different Set Domain Adaptation for Brain-Computer Interfaces: A Label Alignment Approach. *IEEE Transactions on Neural Systems and Rehabilitation Engineering, 28*(5), 1091-1108.

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology, 117*(4), 500.

Izhikevich, E. M. (2006). Polychronization: Computation with Spikes. *Neural Computation, 18*(2), 245-282.

Jiang, J., & Zhai, C. (2007, June). Instance weighting for domain adaptation in NLP. In *Proceedings of the 45th annual meeting of the association of computational linguistics* (pp. 264-271).

Jayaram, V., Alamgir, M., Altun, Y., Scholkopf, B., & Grosse-Wentrup, M. (2016). Transfer learning in brain-computer interfaces. *IEEE Computational Intelligence Magazine, 11*(1), 20-31.

Kasabov, N. K. (2007). *Evolving connectionist systems: the knowledge engineering approach*. Springer Science & Business Media.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images.

Koessler, L., Maillard, L., Benhadid, A., Vignal, J., Felblinger, J., Vespignani, H., & Braun, M. (2009). Automated Cortical Projection of EEG Sensors: Anatomical Correlation via the International 10-10 System. *Neuroimage, 46*(1), 64-72.

Kasabov, N. (2010). Neural Networks Letter: To Spike or Not to Spike: A Probabilistic Spiking Neuron Model. *Neural Networks, 23*(0893-6080), 16019.

Kulis, B., Saenko, K., & Darrell, T. (2011, June). What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *CVPR 2011* (pp. 1785-1792). IEEE.

Kasabov, N. (2012, September). Neucube evospike architecture for spatio-temporal modelling and pattern recognition of brain signals. In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition* (pp. 225-243). Springer, Berlin, Heidelberg.

Kasabov, N., Dhoble, K., Nuntalid, N., & Indiveri, G. (2013). Dynamic Evolving Spiking Neural Networks for On-Line Spatio-and Spectro-Temporal Pattern Recognition. *Neural Networks,* 41, 188-201.

Kasabov, N. (2014). NeuCube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data. *Neural Networks*, 52, 62-76.

Kasabov, N., & Capecci, E. (2015). Spiking neural network methodology for modelling, classification and understanding of EEG spatio-temporal data measuring cognitive processes. *Information Sciences*, 294, 565-575.

Kasabov, N., Scott, N., Tu, E., Marks, S., Sengupta, N., Capecci, E., Othman, M., Doborjeh, M., Murli, N., Hartono, R. & Espinosa-Ramos, J. (2016). Design methodology and selected applications of evolving spatio-temporal data

machines in the NeuCube neuromorphic framework. *Neural Networks*, *78*(2016)), 1-14.

Kasabov, N. K., Doborjeh, M. G., & Doborjeh, Z. G. (2017). Mapping, learning, visualization, classification, and understanding of fMRI Data in the NeuCube evolving spatiotemporal data machine of spiking neural networks. *IEEE transactions on Neural Networks and Learning Systems, 28*(4), 887-899.

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. & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences, 114*(13), 3521–3526.

Kumarasinghe, K., Owen, M., Taylor, D., Kasabov, N., & Kit, C. (2018, May). FaNeuRobot: A Framework for Robot and Prosthetics Control Using the NeuCube Spiking Neural Network Architecture and Finite Automata Theory. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1-8). IEEE.

Koefoed, L., Capecci, E., & Kasabov, N. (2018, July). Analysis of gene expression time series data of ebola vaccine response using the neucube and temporal feature selection. In *2018 International Joint Conference on Neural Networks (IJCNN)* (pp. 1-7). IEEE.

Kasabov, N. K. (2019). *Time-space, spiking neural networks and brain-inspired artificial intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg.

Kumarasinghe, K., Kasabov, N., & Taylor, D. (2020). Deep learning and deep knowledge representation in Spiking Neural Networks for Brain-Computer Interfaces. *Neural Networks*, *121*, 169-185.

Rusu, A. A., Večerík, M., Rothörl, T., Heess, N., Pascanu, R., & Hadsell, R. (2017, October). Sim-to-real robot learning from pixels with progressive nets. In *Conference on Robot Learning* (pp. 262-270). PMLR.

Lomonaco, V., & Maltoni, D. (2017, October). Core50: a new dataset and benchmark for continuous object recognition. In *Conference on Robot Learning* (pp. 17-26). PMLR.

Li, Z., & Hoiem, D. (2017). Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence, 40*(12), 2935-2947.

Li, H., Parikh, N. A., & He, L. (2018). A novel transfer learning approach to enhance deep neural network classification of brain functional connectomes. *Frontiers in neuroscience, 12*, 491.

McCloskey, M., & Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation* (Vol. 24, pp. 109-165). Academic Press.

McClelland, J. L., McNaughton, B. L., & 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), 419.

Mihalkova, L., Huynh, T., & Mooney, R. J. (2007, July). Mapping and revising markov logic networks for transfer learning. In *Aaai* (Vol. 7, pp. 608-614).

Maltoni, D., & Lomonaco, V. (2019). Continuous learning in single-incremental-task scenarios. Neural Networks, 116, 56-73.

Mohseni, M., Shalchyan, V., Jochumsen, M., & Niazi, I. K. (2020). Upper limb complex movements decoding from pre-movement EEG signals using wavelet common spatial patterns. *Computer methods and programs in biomedicine, 183*, 105076.

Neves, A. C., González, I., Leander, J., & Karoumi, R. (2017, July). A new approach to damage detection in bridges using machine learning. In *International Conference on Experimental Vibration Analysis for Civil Engineering Structures* (pp. 73-84). Springer, Cham.

Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering, 22*(10), 1345-1359.

Pan, S. J., Tsang, I. W., Kwok, J. T., & Yang, Q. (2010). Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks, 22*(2), 199-210.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., & Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural Networks, 113*, 54-71.

Raina, R., Battle, A., Lee, H., Packer, B., & Ng, A. Y. (2007, June). Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning* (pp. 759-766).

Song, S., Miller, K. D., & Abbott, L. F. (2000). Competitive Hebbian Learning Through Spike-timing-dependent Synaptic Plasticity. *Nature Neuroscience, 3*(9), 919-926.

Schrauwen, B., & Van Campenhout, J. (2003). BSA, A Fast and Accurate Spike Train Encoding Scheme. In *Proceedings of the International Joint Conference on Neural Networks* (Vol. 4, pp. 2825-2830). IEEE Piscataway, NJ.

Talairach, J., & Tournoux, P. (1988). Co-planar Stereotaxic Atlas of the Human Brain. 3- Dimensional Proportional System: An Approach to Cerebral Imaging. *Thieme Medical Publishers*. New York.

Thorpe, S. J. (1990). Spike Arrival Times: A Highly Efficient Coding Scheme for Neural Networks. *Parallel Processing in Neural Systems,* 91-94.

Thorpe, S., & Gautrais, J. (1998). Rank order coding. *Computational Neuroscience, 13*, 113–119.

Taylor D, Scott N, Kasabov N, Capecci E, Tu E, Saywell N, Chen Y, Hu J, Hou ZG. (2014, July). Feasibility of neucube snn architecture for detecting motor execution and motor intention for use in bciapplications. In *2014 International Joint Conference on Neural Networks (IJCNN)* (pp. 3221-3225). IEEE.

Tu, E., Kasabov, N., & Yang, J. (2016). Mapping temporal variables into the neucube for improved pattern recognition, predictive modeling, and understanding of stream data. *IEEE Transactions on Neural Networks and Learning Systems, 28*(6), 1305-1317.

Wysoski, S. G., Benuskova, L., & Kasabov, N. (2010). Evolving Spiking Neural Networks for Audiovisual Information Processing. *Neural Networks, 23*(7), 819-835.

Wu, D., Lawhern, V. J., Hairston, W. D., & Lance, B. J. (2016). Switching EEG headsets made easy: Reducing offline calibration effort using active weighted

adaptation regularization. *IEEE Transactions on Neural Systems and Rehabilitation Engineering, 24*(11), 1125-1137.

Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. *Journal of Big data, 3*(1), 9.

Zadeh, L. A. (1965). Fuzzy sets. *Information and control, 8*(3), 338-353.

Zenke, F., Poole, B., & Ganguli, S. (2017, July). Continual learning through synaptic intelligence. In *International Conference on Machine Learning* (pp. 3987-3995). PMLR.

Appendix A Talairach Mapping

The EEG mapping into the NeuCube framework was performed according to the 3D coordinates in the Talairach space as presented in Table A-1.

Table A-1 Anatomical locations of cortical projections from (Koessler, et al., 2009)

Labels	Talairach coordinates		
	x (mm)	y (mm)	z (mm)
Fp1	-20	60	10
Fpz	0	60	10
Fp2	20	60	10
AF7	-40	40	30
AF3	-20	60	20
AF4	20	60	20
AF8	40	40	30
F7	-50	30	20
F5	-40	30	20
F3	-30	30	40
F1	-10	30	50
Fz	0	30	50
F2	10	30	50
F4	30	30	40
F6	40	30	20
F8	60	30	20
FT7	-50	10	0
FC5	-50	10	50
FC3	-40	10	50
FC1	-20	10	60
FCz	0	10	60

FC2	20	10	60
FC4	40	10	50
FC6	50	10	50
FT8	50	10	0
T7	-50	-10	0
C5	-40	-10	50
C3	-30	-10	50
C1	-20	-10	60
Cz	0	-10	60
C2	20	-10	60
C4	30	-10	50
C6	40	-10	50
T8	50	-10	0
TP7	-60	-40	0
CP5	-50	-30	20
CP3	-30	-20	60
CP1	-20	-20	60
CPz	0	-20	60
CP2	20	-20	60
CP4	30	-20	60
CP6	50	-30	20
TP8	60	-40	0
P7	-50	-50	0
P5	-40	-50	20
P3	-30	-40	40
P1	-10	-40	50
Pz	0	-60	60
P2	10	-40	50
P4	30	-40	40
P6	40	-50	20

P8	50	-50	0
PO7	-40	-60	0
PO3	-20	-50	30
POz	0	-80	30
PO4	20	-50	30
PO8	40	-60	0
O1	-30	-80	10
Oz	0	-80	20
O2	30	-80	10
TP9	-60	-30	20
TP10	60	-30	20

Appendix B Transfer Learning Study

B.1 Default Parameter settings for NeuCube Model

Table B-1 default parameter setting of the NeuCube model

Method	**Parameter**	**Description**	**Default value**
Encoding	Threshold		0.5
LIF	Firing Threshold	Threshold voltage value to emit a spik	0.5
	Refractory Time	The time period during which a neuron rests after firing	6
	Potential leak rate		0.002
Unsupervised learning	STDP rate	Determines positive synaptic modifications	0.01
Supervised learning	Mod		0.8
	Drift		0.005
	k	The number of nearest neighbours	3

B.2 Statistics Analysis of Output Layer After Neuron Aggregation

Table B-2 The the number of output neurons before and after aggregation

Transfer Learning Across Tasks in One Subject					
Neuron aggregation TrSNN-CP-NG					
	Class 1	Class 2	Class 3	Class 4	Total
Number of neurons before aggregation	15	16	15	15	61
Number of neurons after aggregation	11	14	12	14	51
Neuron aggregation TrSNN-OP-NG					
	Class 1	Class 2	Class 3	Class 4	Total
Number of neurons before aggregation	15	16	15	15	61
Number of neurons after aggregation	8	13	11	14	46
Transfer Learning Across Multiple Subjects					
Neuron aggregation TrSNN-CP-NG					
	Class 1	Class 2	Class 3	Class 4	Total
Number of neurons before aggregation	27	20	23	20	90
Number of neurons after aggregation	23	20	23	20	86
Neuron aggregation TrSNN-OP-NG					
	Class 1	Class 2	Class 3	Class 4	Total
Number of neurons before aggregation	27	20	23	20	90
Number of neurons after aggregation	16	12	13	11	52