UNIVERSITY OF
CAMBRIDGE

Encoding parameter and structural efficiency in deep learning

Simeon Spasov

Robinson College

This dissertation is submitted for the degree of Doctor of Philosophy

Abstract

The development of deep learning has led to significant performance gains on a variety of tasks in a diverse set of application areas spanning computer vision, natural language processing, reinforcement learning, generative modelling and more recently relational learning from graph-structured data. The main reason for this success is an increase in the availability of computational power, which allows for deep and highly parameterized neural network architectures which can learn complex feature transformations from raw data. The high representational power of deep neural networks, however, often comes at the cost of high model complexity which refers to the high parameterization, as well as memory and computational burden associated with deep learning. In this thesis, I rely on parameter-efficient neural operators, appropriate modelling assumptions about the data and inductive biases on network structure to propose simpler neural network models in several application areas. For each of the application areas I work in, I use a combination of these principles of efficiency to design novel approaches. First, within the context of medical image processing, I make the observation that spatially aligned neuroimages exhibit fewer degrees of freedom than natural images, which justifies the use of lower capacity convolutional operators. I achieve this by applying parameter-efficient convolutional variants. I demonstrate state-of-the-art results on early Alzheimer's prediction while using up to 125 times fewer parameters and over 17 times fewer multiply–accumulate operations. Similar conclusions are also reached for an unsupervised method for neuroimages designed to identify subject subtypes. Second, I set out to alleviate the challenge of training parameter-efficient deep models from scratch. This can reduce the infeasibility of training deep models on resource-constrained "edge" devices. The proposed method is based on a simplifying assumption about network architecture, that is parameter independence, which allows to model the problem in the context of combinatorial multi-armed bandits. The method can dynamically, that is during training, identify a high-performing compact subnetwork within an overparameterized model while adhering to a predefined memory utilization budget. This is achieved by associating a saliency metric with each neuron, which is then used to drive parameter activation akin to a gating mechanism, while simultaneously learning the parameters. As a result, the computational and memory burden during both training *and* inference of deep neural networks is significantly reduced. Finally, I present a deep probabilistic model for

learning unsupervised node and community embeddings in dynamic graphs. I introduce structural inductive biases about the edge formation mechanism based on the inherent community structure of networks. Further, I also assume smooth temporal evolution of both the nodes and the communities inspired by the lack of disruptive events in the data. I present a parameter-efficient implementation of the method which outperforms state-of-the-art graph convolutional networks on a variety of dynamic predictive tasks.

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text. It is not substantially the same as any that I have submitted, or am concurrently submitting, for a degree or diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. I further state that no substantial part of my dissertation has already been submitted, or is being concurrently submitted, for any such degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. This dissertation does not exceed the prescribed limit of 60 000 words.

Simeon Spasov
May, 2021

Acknowledgements

This thesis is the result of the most academically challenging period of my career so far, and would not have been possible without the support of many people. First, I would like to thank my supervisor, Prof. Pietro Liò for the opportunity to be part of his group, his research guidance, and all the academic and personal help he has given me. I would also like to express my gratitude to the Department of Computer Science and Technology for the wonderful environment they have created. I feel very fortunate, happy and proud I was part of the Computer Laboratory for a few short years. Further, I would like to thank Prof. Nicola Toschi and Dr. Luca Passamonti for a very fruitful and long collaboration I was able to learn so much from. Also, I would like to thank Prof. Jian Tang for supervising me during my visit at the Montreal Institute for Learning Algorithms. Finally, I would like to thank my family and friends for providing me with all the support I needed.

Contents

Chapter 1

Introduction

Ever since the AlexNet [117] deep convolutional neural network won the ImageNet [190] classification competition, deep learning (DL) has seen rapid development. One key component that enabled the explosion of deep learning was the availability of computational power, and more specifically GPUs, paired with highly optimized implementation of parallelized operations, which facilitate the training of large models. Another contributing factor was the increasing accessibility of large, highly dimensional datasets comprising millions of samples. Significant research efforts have been put towards developing new methods, and as a consequence, extraordinary progress has been made on a variety of increasingly complex tasks within the field of computer vision [74, 90, 185, 117], natural language processing [218, 231, 40] and reinforcement learning [151, 152, 150]. For example, recent milestone achievements include deep learning systems outperforming humans on visual reasoning tasks (e.g. classification) of natural images [88], and even outperforming human clinical experts on breast cancer prediction [147], as well as dominating strategy games like Go [200] and StarCraft [235] against the world's top players. These examples represent only a cursory glance at what is currently possible and are by far not exhaustive.

The success of state-of-the-art deep learning models stems from their ability to build rich representations from data and extract complicated dependencies between the input features. Often, this comes at the cost of high *model complexity*, which within the context of this thesis, refers to the high parameterization, as well as memory and computational burden associated with deep learning. Although recent research indicates that overparameterization can benefit training dynamics by facilitating convergence of gradient decent [3, 163], it hampers the application of deep learning in certain scenarios. One issue of overparameterization is poor performance generalization to unseen test samples. Sometimes overparameterization can be so severe that deep neural networks would be able to memorize arbitrary patterns in the training data [252]. This means models tend to need millions of samples to train well, or they would overfit to the training data.

Collecting such vast datasets is not always possible, for example in the biomedical sphere - it can be *time-consuming* and *expensive* to collect patient data as well as raise *ethical concerns*. Another problem of overparameterization comes with additional memory and computational requirements during model training and inference. In fact, the established paradigm requires that user data is sent to dedicated servers which run the deep learning models and communicate back a prediction. This can leave user data compromised and exposed to certain attacks, and is in general considered a significant data privacy and protection challenge [106]. Only recently have we started seeing a move towards *model simplicity* - a development of memory and computationally efficient methods appropriate for cost-conscious settings.

One research direction which has helped reduce model complexity is the development of (1) *parameter-efficient neural operators*. Depthwise separable convolutions [29] are an early attempt at reducing computational burden and have been successfully applied in mobile devices [99]. Since then, convolutions have also been generalized to exploit group symmetries and weight sharing for lower training data requirements [31], reformulated specifically for efficient video processing [83], object detection [220] or pose estimation [216], among others. Another approach to reducing complexity is (2) incorporating appropriate *assumptions about the data* we are trying to model. A well established example is substituting feedforward layers for convolutional kernels in image processing applications as the latter only retain local correlations between features. A more recent example is the development of spherical convolutional networks [49, 32] aimed specifically at processing spherical images by incorporating rotational equivariance properties. Finally, we can also (3) discover strong *inductive biases about the structure* of the network architecture itself. The Lottery Ticket Hypothesis [64], and its derivatives for deeper models [67, 22] and BERT networks [22], exemplify this trend by using iterative pruning to find small subnetwork structures that can train to similar accuracy as a more complex network in a commensurate number of steps. Attempts from all these research directions (1), (2) and (3) serve as evidence of a move towards developing *simpler neural models*, which retain strong performance on the data structures they are designed to work on despite lower memory and computational requirements.

The need for simpler models can be motivated from both biological and philosophical points of view. Recall that one disadvantage of overparameterized, hence unnecessarily complex models, is data-inefficiency and difficulties generalizing to unobserved data points. On the other hand, human babies do not need millions, or even hundreds, of examples to learn to visually recognize objects from daily life. This is because the efficient baby learner is *born* with dedicated brain areas which specialize in a certain task and are highly

optimized to perform it. For instance, the visual cortex contains individual cells that are sensitive to the orientation of input stimuli [101]. This improves the capability to recognize certain visual patterns. The specialized structure of the visual cortex can be seen as an *inductive bias* [149]. Evolution has found a way to inject a preference for a model architecture which can significantly improve generalization and data efficiency. The need for inductive biases can be seen through the lens of the *no free lunch theorem* [241] which states that every classification algorithm, regardless of its level of sophistication, has the same error rate, averaged over all data generating distributions, on unobserved data points. Hence, no learning algorithm is in general better than any other. For this reason, it is important to understand the specific problem we are trying to solve and design machine learning models by injecting prior knowledge about the problem. Finally, a strong argument for achieving model simplicity is Occam's razor, which can be seen as an abductive reasoning heuristic during model design. In simple terms, Occam's razor prefers the simplest explanation for a set of observations among all other hypotheses.

In this dissertation, I apply principles of efficiency to propose designs of *simple neural network models* in several application areas. I rely on parameter-efficient operators and introducing appropriate modelling assumptions about the data or network structure. For each application, first I propose a set of inductive biases inspired by the structure of the neural network model or prior knowledge about the data. This forms the first step towards achieving model simplicity. Then, when appropriate, I propose a parameter-efficient solution. In Chapter 3, I propose a parameter and computationally efficient neural network architecture for neuroimaging applications. The main novelty of the work stems from the observation that neuroimages are spatially aligned to a common reference, hence each feature always occupies the same location in space. In turn, we can reduce the needs for representational capacity, especially compared to natural images which exhibit greater variety. I achieve this by applying a parameter-efficient version of the common convolutional operator. In Chapter 4, I propose an algorithm applied during training which can identify the most salient subnetwork within an overparameterized neural network model. The procedure is based on a simplifying prior assumption about network architecture, that is parameter independence. Finally, in Chapter 5, I propose a structured probabilistic model for dynamically evolving graphs. I design a generative process which incorporates an inductive bias about the naturally emerging community structure of real-world networks as well as an assumption of smooth temporal evolution void of disruptive events. A parameter-efficient implementation of the proposed method is also provided. Certain model parameters are shared and the parameterization of the model is kept independent of the number of time points in the data.

In the remaining sections of this Chapter, first I specify in more detail the research goals I address in the thesis. Then, I provide an overview of the structure of the document. This includes summaries of each chapter as well as a graphical abstract. Finally, I enumerate my published works and clarify the contributions of each of my collaborators.

1.1 Research goals

Goal 1 *Demonstrate the applicability of parameter-efficient convolutional operators on spatially co-registered neuroimages.*

A key feature in neuroimaging is that we know a priori the shape and location of the object a neuroimage would contain. On the other hand, we can have no such knowledge about a natural image. For this reason, it is common practice in neuroimaging to map all brain images to a standard template to remove all shape and size variability, hence facilitate the study of intersubject brain function. This normalization procedure is called image co-registration and drastically reduces the need for representational capacity in a learning algorithm. As a consequence, we can place an inductive bias for lower model complexity and achieve it via *parameter-efficient* convolutional operators. The advantages of this are two-fold: first, lower computational cost can speed up medical imaging research turnaround time, and second, it democratizes research by reducing the need for computational power.

Goal 2 *Reducing neural network overparameterization by imposing an inductive bias on parameter independence.*

The majority of studies so far have addressed the challenge of reducing computational effort during *inference*. In this thesis, I also attempt to enable training overparameterized deep neural networks on memory- and computationally constrained edge devices. To achieve this goal, I propose a reinforcement learning-based approach which only uses a subset of all model parameters at each forward pass through a batch of data. The proposed algorithm can simultaneously associate a saliency metric with each neuron, which is used to drive channel activation akin to a gating mechanism, as well as learn the model parameters. The algorithm is based on an inductive bias about model parameter independence, which makes the computation of parameter saliency tractable.

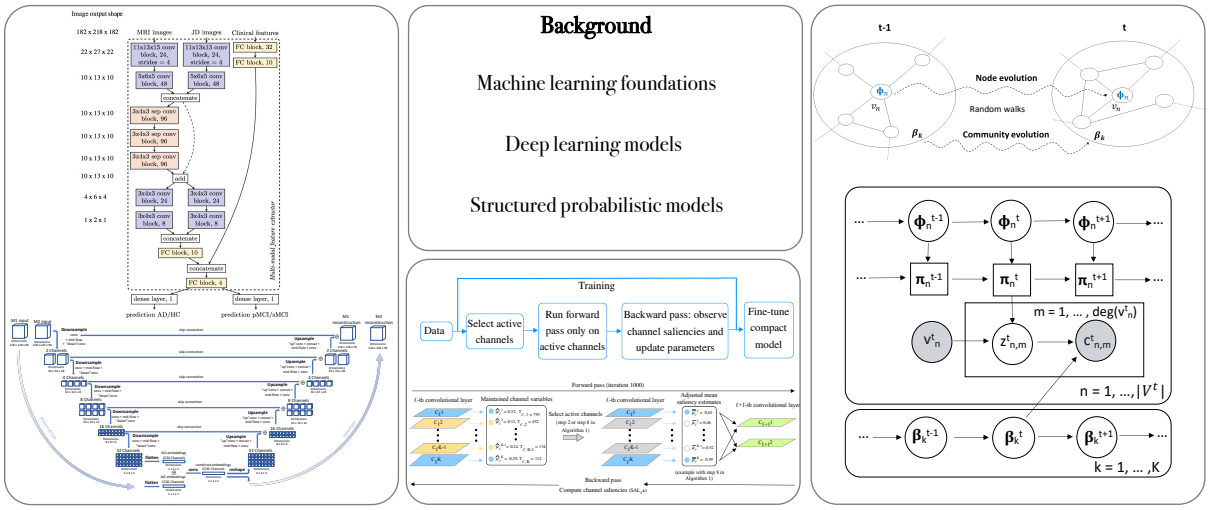
Goal 3 *A parameter-efficient generative model for dynamic graphs based on structural inductive biases about community structure and smooth temporal evolution.*

The majority of graph representation learning methods have focused solely on static non-evolving graphs, whereas many real-world networks exhibit complicated temporal dynamics. I develop a deep probabilistic model for learning evolving node and community embeddings. First, I encode prior knowledge about the tendency of nodes to form communities in networks. I introduce a *hidden variable to represent community assignment* for each node which facilitates learning the mechanism of edge formation. Second, I assume *smooth temporal evolution* of both the node and community representations. Finally, I design an efficient variational procedure for learning the model parameters. The *parameterization is kept low* by taking specific precautions, e.g. resorting to amortized inference, parameter sharing, and using implementation mechanisms which make the the number of model parameters *independent* of the number of time steps.

1.2 Thesis structure

Each of the research questions posed in Section 1.1 *Research goals* is addressed separately in a dedicated chapter as depicted in Fig. 1.1. The Research goals, and the respective chapters addressing them, are conceptually connected with respect to the overarching goal of achieving efficiency although the contexts of application are different.

Chapter 2: Background. I begin by giving the necessary theoretical foundation required to conduct the research presented in the main chapters. First, I present the basics of feedforward and convolutional neural networks with a focus on *parameter-efficient* variants, such as separable and grouped convolutions. More specifically, I discuss how certain neural network design choices can impact parameterization as well as *model capacity*, and as a result - the generalization of model performance to unseen data. Then, I proceed to introduce the principle of *maximum likelihood* and a variety of regularization and training procedures, necessary for optimizing neural networks. Finally, I present *structured probabilistic modelling* as a way to design efficient deep generative models while incorporating prior knowledge about the data generative mechanism.



Chapter 3
Parameter-efficient neural network design

Chapter 4
Efficient training on-device

Chapter 5
Inductive biases in dynamic graphs via probabilistic modelling

Figure 1.1: A graphical abstract of the thesis structure. After providing the necessary theoretical background, I propose incorporating appropriate modelling assumptions with parameter-efficient implementations as a way to reduce model complexity. In Chapter 3, I demonstrate the efficacy of using computationally and memory efficient operators as an alternative to the standard convolution in neuroimage processing. The design choice is supported by observations about the structure of the input data. Then, in Chapter 4, I introduce a framework which can identify the most salient subnetwork within an overparameterized model during training using a reinforcement learning algorithm and a simplifying assumption about parameter dependency. Lastly, in Chapter 5, I explore *incorporating prior knowledge* about community structure and temporal dynamics in a probabilistic generative model for dynamic graphs. All notations used in this figure are explained in detail in the respective chapters.

Chapter 3: Parameter-efficient deep learning in neuroimaging. In this chapter, I argue that specific design precautions are advantageous when applying neural networks on neuroimages. More specifically, we should exploit the knowledge that such data comes in a well known and predefined format with lower representational capacity needs than natural images. I advocate for the use of convolutional operators of lower computational and memory cost, such as separable and grouped convolutions, as well as the utilization of skip connections. The chapter is organised in two sections. In section 3.1., I tackle the prognostication task of Mild Cognitive Impairment to Alzheimer's disease classification and achieve *state-of-art results*. In addition, I demonstrate that the proposed deep learning model is *interpretable*, in that it recognizes brain areas clinically associated with the development of Alzheimer's disease as highly important for the classification task. In section 3.2, I present the design of a deep 3D multimodal autoencoder with high image

reconstruction fidelity, and capable of producing subject embeddings useful for population stratification.

Chapter 4: On-device neural network training. In this chapter, I tackle Research Goal 2, that is dynamically identifying a high-performing subnetwork within an overparameterized neural network model while simultaneously learning its parameters. I describe a set of modelling assumptions about network architecture which allow to formulate the problem in the context of *combinatorial multi-armed bandits*, that is selecting k out of n resources (or neurons/convolutional channels). Then, I propose a reinforcement learning approach based on the combinatorial upper confidence bound algorithm, which does not introduce any additional parameterization, to solve it. I validate the methodology on a range of datasets and neural network architectures (both purely sequential and skip-connection based). In all cases, the proposed framework is able to outperform a random baseline, and achieves significant parameter and floating point operation reductions, e.g. often a $10\times - 20\times$ decrease in parameterization with minimal degradation in accuracy. It is *essential* to stress that the proposed method is *fully integrated within the training procedure* as opposed to other works, which treat the goal of reducing model size and/or computational effort as a post-processing technique, and are aimed only at inference.

Chapter 5: Inductive biases in probabilistic modelling for dynamic graphs. In Chapter 5, I present a deep probabilistic model for learning evolving node and community representations. I incorporate prior knowledge into the data generative mechanism by exploiting ideas from network science presenting evidence that *real-world networks form communities* consist of densely connected nodes. Further, I also incorporate an assumption about *smooth temporal dynamics*, that is the lack of any disruptive events in the data. All probability distributions in the model are parameterized by neural networks trained via variational inference. A parameter-efficient implementation is provided by using amortized inference and parameter sharing. The efficacy of the method is validated against state-of-the-art graph neural networks on dynamic link prediction and dynamic community detection, as well as the *novel task* of predicting community-scale dynamics.

Chapter 6: Conclusion and future work. I conclude this thesis by summarizing the main contributions of my dissertation once more and providing several avenues for future research. These include producing compact and "light" unimodal student networks from deep multi-modal teachers; applying efficient on-device training in federated learning and for on-device personalization; extending the unsupervised deep generative model introduced in Chapter 5 to a multi-graph learning setting and validating it on fMRI data.

1.3 List of publications and contributions

Here, I provide a list of the publications I have produced as a result of my PhD studies. The majority of them are a consequence of the contributions related to the Research Goals. Other are a result of my duties as a supervisor of Part III and MPhil students as well as my involvement in the PropagAgeing European Project. The publications are enumerated in the order in which they were completed.

[208] **Spasov S.**, Passamonti L, Duggento, A. and Liò P. and Toschi N., A Multi-modal Convolutional Neural Network Framework for the Prediction of Alzheimer's Disease. *40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2018.* Presented as an *oral talk*.

[204] **Spasov S.**, Passamonti L, Duggento A., Liò P. and Toschi N., A parameter-efficient deep learning approach to predict conversion from mild cognitive impairment to Alzheimer's disease., 2019, p. 276-287, *Neuroimage*. Also presented at *Annual Meeting of the Organisation of Human Brain Mapping 2019 (OHBM).*

[206] **Spasov S.** and Liò P., Dynamic Neural Network Channel Execution for Efficient Training, *2019, 30th British Machine Vision Conference (BMVC)*. Also presented at the *33rd Annual Conference on Neural Information Processing Systems (NeurIPS), Workshop on Energy-Efficient Machine Learning (EMC2), 2019 [207].*

[140] Maj C., Azevedo, T., Giansanti, V, Borisov, O., Dimitri, G.M., **Spasov, S.**, Liò, P., and Merelli, I., Integration of machine learning methods to dissect genetically imputed transcriptomic profiles in Alzheimer's Disease, 2019, *Frontiers in Genetics*

[222] Taylor D., **Spasov S.** and Liò, P, Co-Attentive Cross-Modal Deep Learning for Medical Evidence Synthesis and Decision Making, *33rd Annual Conference on Neural Information Processing Systems (NeurIPS), Machine Learning for Health (ML4H) Workshop, 2019.*

[70] Glass, S., **Spasov, S.**, and Liò, P., Curvature-guided Pruning of High-performance Neural Networks Using Ricci Flow, *International Conference on Machine Learning (ICML), Workshop on Automated Machine Learning (AutoML), 2020*

[42] Dimitri G.M., **Spasov S.**, Duggento A., Passamonti L., Toschi N., Liò P., Unsupervised stratification in neuroimaging through deep latent embeddings. *42nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2020.* Presented as an *oral talk*.

[176] Pirazzini, C., Azevedo, T., Baldelli, L., Bartoletti-Stella, A., Calandra-Buonaura, G., Dal Molin, A., Dimitri, G.M., **Spasov S.**, Doykov, I., Gómez-Garre, P., Hägg, S.

and Hällqvist, J., and others., 2020, A Geroscience approach for Parkinson's Disease: conceptual framework and design of PROPAG-AGEING project. *Mechanisms of Ageing and Development.*

[43] Dimitri G.M., **Spasov S.**, Duggento A., Passamonti L., Toschi N., Liò P., Multimodal image fusion via deep generative models. *Information Fusion* (In submission)

[205] **Spasov, S.**, Di Stefano, A., Liò, P, and Tang, J., GRADE: Graph Dynamic Embedding, https://arxiv.org/abs/2007.08060

I also provide further clarifications on the contributions of my co-authors and collaborators, as well as details on currently unpublished research, which I have contributed to.

- I took part in the European Project Propag-Ageing by producing models for predicting brain ageing from structural MRI data of Parkinson's and healthy subjects. I presented my work at meetings in London (2018), Cambridge (2019) and Bologna (2019).
- For the publication [140], whose first author is T. Azevedo, I contributed with discussions on the methodology and write-up of the paper.
- The workshop papers with S. Glass and D. Taylor are a result of their Part III Physics/MPhil in Advanced Computing theses completed under my (co-)supervision.
- The first authorship of the publication [42] and the manuscript [43] is *shared* between G.M. Dimitri and myself. My main contribution was developing the method and its software implementation. G.M. Dimitri contributed by running experiments and data analysis. The papers were written by equally splitting the workload.
- N. Toschi and L. Passamonti provided substantial support for all publications in which they are co-authors. More specifically, they contributed with experimental suggestions, clinical interpretations of obtained results and improving the textual presentation of the papers.
- Jian Tang was my advisor during my internship at the Montreal Institute for Learning Algorithms (MILA) and provided ideas about the research direction of my project, which culminated in [205].

The publications associated with Research Goal 1 include [208, 204, 43, 42]. In addition, as a daily supervisor of D. Taylor, I helped guide his masters project on *efficiently* integrating several input modalities in a deep learning-based classification system by

exploiting the cross-correlations in the inputs using an attention mechanism. The work outperforms previously published state-of-the-art unimodal methods while being over 50% more parameter-efficient, and was published as a paper in the *Machine Learning for Health (ML4H) Workshop* associated with the NeurIPS 2019 conference [222]. Research Goal 2 resulted in two publications [206, 207]. I also supervised another student, S. Glass, on a topic related to this research goal. Specifically, his thesis borrows concepts from physics, such as Ricci curvature, to remove unimportant connections from the computational graphs of neural networks. The developed method was published in the *Workshop on Automated Machine Learning* collocated with ICML 2020 [70].

Chapter 2

Background

I begin the Background by outlining the philosophical motivation of the thesis. Then, I proceed with summarising the basics of related work. A large part of the Background chapter was inspired by the *Deep Learning book* [73]. The main chapters include separate tables of notations to ease the readability of the document.

2.1 Modulating complexity by priors, structure and parameter efficiency.

The key goal of machine learning is to train a model on a *training set* and have it perform well on previously unseen data, that is a *test set*. The ability of an algorithm to achieve this goal is called generalization. The performance of the algorithm is quantified by some measure, for example the cost function, which is referred to as the *error*. The generalization capabilities of a given model are measured by the *generalization gap*, that is the difference between the test and train errors. Very poor training set performance is known as underfitting, whereas a large generalization gap is known as overfitting (test error > train error). Refer to Fig. 2.1 for a visualization. Navigating between these two states is done by altering the model's *capacity*, that is the ability of the model to fit a wide variety of functions.

A model with higher complexity, e.g. model parameters, usually also has higher capacity. It is essential to *modulate the capacity of a model with certain design choices* in order to fit the necessary complexity for the problem we are solving. A machine learning algorithm performs best when its capacity is appropriate for the complexity of the task and the availability of data. In this thesis, I use a variety of methods to modulate model complexity by introducing inductive biases based on data assumptions or model structure, or simply employing parameter-efficient neural operators. Here, I present these approaches as well as their connection to the work done in this thesis.

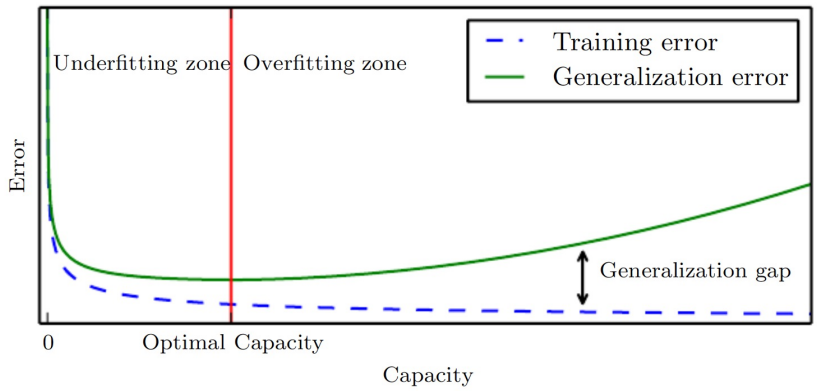


Figure 2.1: The figure illustrates the relationship between model capacity and error. When the model capacity is too low, both the training and test errors are high (underfitting). As capacity is increased, training error decreases, but too high of an increase leads to a deterioration in test set performance and a large generalization gap (overfitting). The red line separates the underfitting and overfitting zones, and depicts the point where model capacity matches the complexity of the machine learning task.

- *Parameter-efficient neural operators.* Reducing the model parameters is the most clear way to also decreasing model complexity, as well as decreasing computational and memory requirements. Consider depthwise separable convolutions, which substitute the conventional convolution with a sequential application of a depthwise and a pointwise convolution [29]. An equivalent depthwise convolutional layer comprises a fraction of the parameters, especially if working with 3D images as I do in Chapter 3. In the following section of the Background, I provide a quantitative analysis of the memory and computational savings provided by this convolutional operator. Depthwise separable convolutions achieve parameter efficiency by decomposing the conventional convolution in two steps. Structured Convolutions [12] build on this concept by decomposing the convolutional operation into sum-pooling followed by a convolution with significantly lower complexity and fewer weights. There are other methods which attempt to decompose convolutions with SVD [253], spatial SVD [104], non-iterative low-rank decomposition [219], CP decomposition [120] and tensor-train decomposition [168, 211, 244]. T. Cohen and M. Welling propose Group Equivariant Convolutional Networks [31], which heavily exploit weight sharing by reformulating conventional convolutions to incorporate equivariance to more general transformations (e.g. rotations and reflections) in addition to translation. The authors achieve better generalization at the same level of parameterization. I also resort

to weight sharing in Chapter 3 where I jointly optimize two objectives which share common underlying factors, hence similar transformations are appropriate. Some strong examples of parameter efficiency from computer vision applications include EfficientDet with a $4\times - 9\times$ smaller model than competitors in object detection [220], as well as HRNet which achieves better performance with comparable or fewer parameters than competitors in pose estimation [237], among other approaches. It is interesting to note that many of these proposed efficient operators can be viewed as *imposing structure* on the convolutional kernels to restrict their degrees of freedom.

- *Data assumptions.* We can use prior knowledge about our data to make educated assumptions about the structure of our deep models. For example, consider that features in planar images are defined by local correlations of activation values. In addition, for many vision applications, the presence itself instead of the exact location of a feature ought to drive prediction behaviour. These data-driven insights led to the understanding that shift-invariant operators, for example convolutions, would be most appropriate for natural images, which also meant that weight sharing can be used for processing each input value [123]. This observation pioneered many advances in computer vision by improving generalization, as well as reducing memory and computational costs. Efficient neural operators for spherical images, commonly used in omnidirectional vision for drones, robots, and autonomous cars, as well as weather and climate modelling, have a different set of desirable properties. A direct application of convolutional networks to a planar projection of a spherical image is ineffective because translational weight sharing is inappropriate for such data. Instead, Spherical CNNs introduce a convolutional variant which is rotation-equivariant, a much more appropriate assumption for analysing spherical signals [49, 32]. Consider the effect of spatial alignment of planar images, such that a feature always occupies the same location in space. This reduces the degrees of freedom of the input data, which in turns reduces the need for capacity in a model. For this reason, in Chapter 3, I decide to heavily rely on depthwise separable convolutions. This choice also benefits from drastic reductions in compute and memory requirements, hence speeding up experimental iterations and democratising medical research. Admittedly, explicitly taking into account the symmetries present in medical images would be even more appropriate to achieve an efficient and simple model. Finally, it is worth mentioning the importance of choosing appropriate data priors for probabilistic models as an example of a reasonable assumption about the data, and its impact on generalization. This topic, and its relation to Chapter 5, is discussed in the Background in section 2.4 Structured probabilistic modelling.

- *Inductive biases on model architeture.* Here, I group methods which impose structure

on the model *not* driven by data assumptions. As an example, consider the lottery ticket structure identified following the procedure outlined in [64]. A very high proportion of the weights in the lottery ticket will be 'masked out'. In effect, a lottery ticket simply accelerates the trajectory towards 0 of these weights during the training process. Hence, a lottery ticket applied prior to training a compact model encodes an inductive bias on which parameters contribute to performance and which are obsolete. In Chapter 4, I introduce a parameter independence assumption driven by a need for efficient computation. The implications of this assumption, its relation to pruning and an overview of the Lottery Ticket Hypothesis are all provided in Chapter 4.

In the remainder of this chapter I provide all necessary background by building from the most basic element of deep models, the individual neuron. Then, I provide a comprehensive overview of convolutional layers with a special focus on depthwise separable convolutions and a quantitative analysis of their compute and memory costs. This analysis is essential for understanding the efficiency brought by work done in Chapter 3 and Chapter 4. I also cover more generic aspects of machine learning, such as the mathematical principles behind training and optimisation of neural networks, regularisation and hyperparameter tuning. Finally, I present structured probabilistic modelling and how inductive biases can be incorporated in this framework. I also provide the formulation of variational autoencoders, which form the basis of my work in Chapter 5.

2.2 Artificial neural network designs

In this section, I present the two quintessential deep learning models, namely fully connected and convolutional neural networks, both of which are represented as acyclical directed graphs of computational units called *neurons*. Both fully connected and convolutional models provide an approximation to a function and take as an input a real-valued vector $\mathbf{x}$. Artificial neural networks (ANNs) define a mapping $\mathbf{y} = NN(\mathbf{x}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ represent the parameters of the neural network. Note that throughout this thesis all vectors are given in ***bold***.

2.2.1 The fully connected layer

The basic structure of a single neuron in a fully connected network model, or a perceptron, is illustrated in Fig. 2.2 (Left). The perceptron computes a linear combination between a set of real-valued parameters and the input feature vector $\mathbf{x}$. Since artificial neural networks usually consist of many neurons, I denote the parameters associated with the j-th neuron as $\boldsymbol{\theta}_j$. The weighted linear combination between the input and the parameter

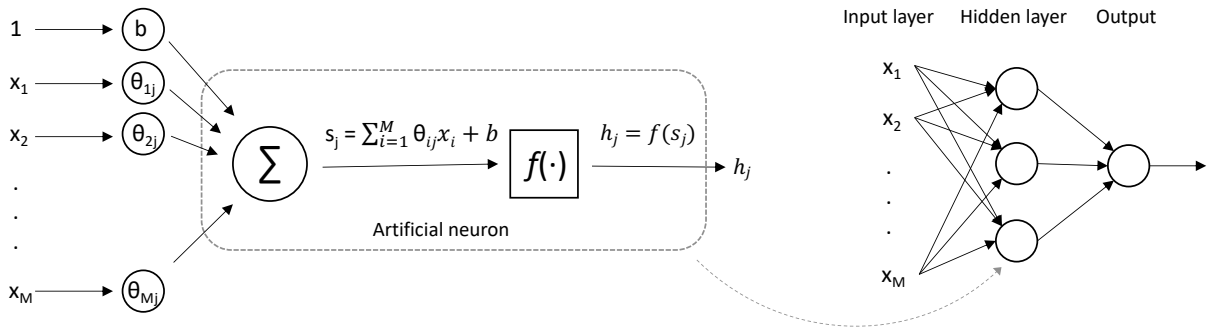


Figure 2.2: Left: The computational model of a single artificial neuron (perceptron). **Right:** A graphical representation of a multilayer perceptron. Each layer comprises many individual artificial neurons.

vector is:

$$s_j = \sum_{i=1}^{M} \theta_{i,j} x_i + b = \boldsymbol{\theta}_j^T \mathbf{x} + b \tag{2.1}$$

It is also common to add a bias value, b, to the weighted sum. To obtain the final activation value of the artificial neuron, a (non-linear) activation function, $f(\cdot)$, can be optionally applied:

$$h_j = f(\theta_j^T \mathbf{x} + b) \tag{2.2}$$

A variety of activation functions have been designed which are applicable in different scenarios. An example is the *sigmoid activation function* which is suitable for modelling Bernoulli random variables, $f(x) = \frac{1}{1+exp(-x)}$, and is used for binary classification tasks as it maps the input to the $[0, 1]$ range. On the other hand, the sigmoid activation suffers from certain drawbacks, e.g. the vanishing gradient problem, which can significantly impede the training process. To remedy this issue, the *rectified linear unit* (ReLU) [71] was introduced, $f(x) = \max(0, \mathrm{x})$, as well as its variants, such as the *leaky ReLU*, the *parametric ReLU* [88] and the *exponential linear unit* (ELU) [30].

The practical success of artificial neural networks, and more specifically deep learning, has come from the utilization of a *very high* number of artificial neurons in a characteristic layered feedforward structure. Each layer comprises many individual perceptrons which receive as an input the vectorised outputs from the previous layer. This is known as a *fully connected* architecture, or a *multilayer perceptron* (MLP), as depicted in Fig. 2.2 (Right). To present the transformation of a single fully connected layer, the neuron

parameters comprising the layer can be organised as row vectors in a matrix $\boldsymbol{\Theta}$:

$$\mathbf{h} = f(\boldsymbol{\Theta}^T\mathbf{x} + \mathbf{b}) \tag{2.3}$$

By stacking several consecutive fully connected layers, we can create a *multilayer perceptron* which performs the transformation $\mathbf{y} = NN(\mathbf{x}; \boldsymbol{\theta})$:

$$\mathbf{y} = f_2(\boldsymbol{\Theta}_2^T f_1(\boldsymbol{\Theta}_1^T\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) \tag{2.4}$$

A natural question to ask is how to determine the number of hidden (intermediate) layers and the number of neurons per layer? Although this question has no concrete answer, the *universal approximation theorem* by Cybenko [34] states that a neural network with a single hidden layer containing a sufficient but finite number of neurons with sigmoidal activations can approximate arbitrarily precisely any bounded continuous real function. This proof by Cybenko, however, does not specify the numbers of neurons needed in this hidden layer, which in many cases could be prohibitively large. More recent results [225, 192] have shown that *deeper networks allow for a more parameter-efficient approximation than shallow networks*, that is the overall number of neurons in the deep networks is lower than their shallow equivalent. This conclusion has also been validated empirically and current neural networks comprise many hidden layers, giving rise to modern *deep learning*.

The ability of deep neural networks to perform complex transformations based on the inputs is essential both for generative models, where we are interested in modelling the data generative distribution itself $p(\mathbf{x}; \boldsymbol{\theta})$, and discriminative models, where we are interested in producing a conditional statistic $\mathbf{y}$ given the input $\mathbf{x}$, that is $p(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta})$.

2.2.2 Convolutional neural networks

In fully connected neural network models, each neuron interacts with each element of the input. This can pose a problem if the input is very highly dimensional as is the case with images, which often comprise millions of pixels, and especially 3D medical images. The high number of input features introduce a high number of model parameters in each neuron, which hurts the statistical generalization ability of the network. To handle high resolution and intricate image data, we need to exploit the inherent *spatial structure* in images and the idea of *locality*. The *convolutional* neural network was introduced specifically to achieve this goal [123].

To explain the basic convolutional operation, I assume the inputs $\boldsymbol{x}$ comprise 2D images of shape $H \times W$ and have C channels, for example red, green and blue intensity

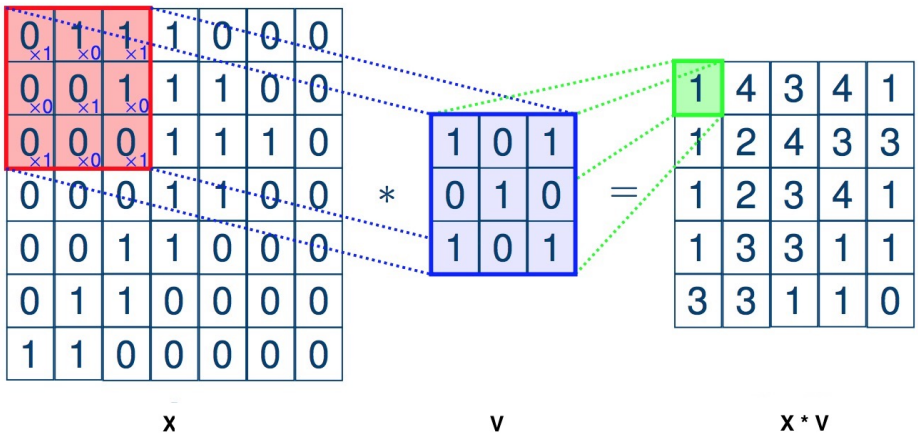


Figure 2.3: An illustration of the convolutional operation. To produce a single output element, the kernel is overlaid on a sub-region of the input image and summed element-wise.

channels at each pixel for colour images. Also, I introduce a 4D kernel tensor $\boldsymbol{V}$ of shape $m \times n \times C \times C'$. Convolving an input image $\boldsymbol{x}$ with the kernel $\boldsymbol{V}$ results in an output activation, also known as a *feature map*, $\boldsymbol{h}$ of size $H' \times W'$ with C' channels. To produce each element of the resulting feature map, $\boldsymbol{h}$, the convolutional operator performs the following computation:

$$h_{a,b,c'} = f(b_{c'} + \sum_{c=1}^{C} \sum_{i=1}^{n} \sum_{j=1}^{m} V_{i,j,c,c'} \cdot x_{a+i-1,b+j-1,c}) \tag{2.5}$$

The convolution works by overlaying the kernel on the input image and computing an element-wise sum between the kernel and image sub-region across all channels to produce a single element in the output feature map (see Fig. 2.3). The full feature map is produced by sliding the kernel across the input image. Similarly to the perceptron model, a bias value is optionally added after spatial convolution and the output is passed through an activation function. In the full convolutional layer, we perform the operation in Eq. 2.5 for all C' output layers.

Convolutional layers exploit three key concepts which are advantageous for deep learning systems applied on images - *sparse interactions, parameter sharing* and *equivariant representations*. Firstly, unlike fully connected layers, where each output unit interacts with every input unit, convolutional layers exploit kernels with a lower dimensionality than the inputs (*sparse interaction*). As a result, the memory requirements to store the model

parameters are significantly lower compared to using fully connected layers. In addition, computations require significantly fewer operations. The resulting parameter-efficiency also improves the generalization capabilities of convolutional models. Second, parameter and computational efficiency are also improved by *parameter sharing* since each element of the kernel is used at every position of the input. Finally, the convolutional operation is *translation equivariant* - if the input is translated, the learned activation maps will spatially shift in exactly the same way. This property is very useful for some image applications, where the behaviour of a convolutional model should not depend on the position of an object in the image.

There are other variants of the convolutional layer. More specifically, in this thesis, I resort to the use of *parameter-efficient variants*, namely the *separable* and *grouped* convolutions. In conventional convolutions, the inputs are mixed spatially and channel-wise in a single step by applying a kernel of size $m \times n \times C$ over all input channels to form a single channel of the output feature map. As a result, C' such kernels are needed to produce all feature map outputs. In contrast, separable convolutions split this procedure in two by first applying a *depthwise* convolution and then a *pointwise* convolution. First, depthwise convolutions are applied over a *single* channel of the input to produce a single channel of the output. This implements feature extraction only *spatially*. Then, the resulting activation maps are stacked together, and C' different 1×1 convolutions are applied to mix the features channel-wise (pointwise convolutions). The full procedure is shown in Fig. 2.4 (b). Depthwise separable convolutions lead to significant parameter-efficiency improvements. A single conventional convolutional layer comprises $m \times n \times C \times C'$ parameters, whereas for depthwise separable convolutions $m \times n \times C + C \times C'$ parameters are required. For equivalent layers producing the same number of output channels, this leads to $\frac{1}{C'} + \frac{1}{m \times n}$ fewer parameters for the separable case. Since usually the number of output channels, C', is significantly larger than the kernel size, $m \times n$, separable convolutions are $\sim m \times n$ times more parameter-efficient. Another useful metric to quantify efficiency is counting the overall number of multiplication and addition operations. For instance, in the conventional convolution, for each element in the output feature map, we need to perform $m \times n \times C \times C'$ multiply-accumulate operations, whereas for the depthwise separable ones: $m \times n \times C + C \times C'$. We can now see the computational speed up is proportional to the reduction in parameterization. A comparison between different convolutional variants in presented in table 2.1. It is essential to note that heavy reliance on separable convolutions allows for very deep models, which have proved to be better performing than more shallow ones, while parameter-efficiency aids in better generalization and computational efficiency.

The grouped convolutions can be seen as a variant of the depthwise convolutions. The

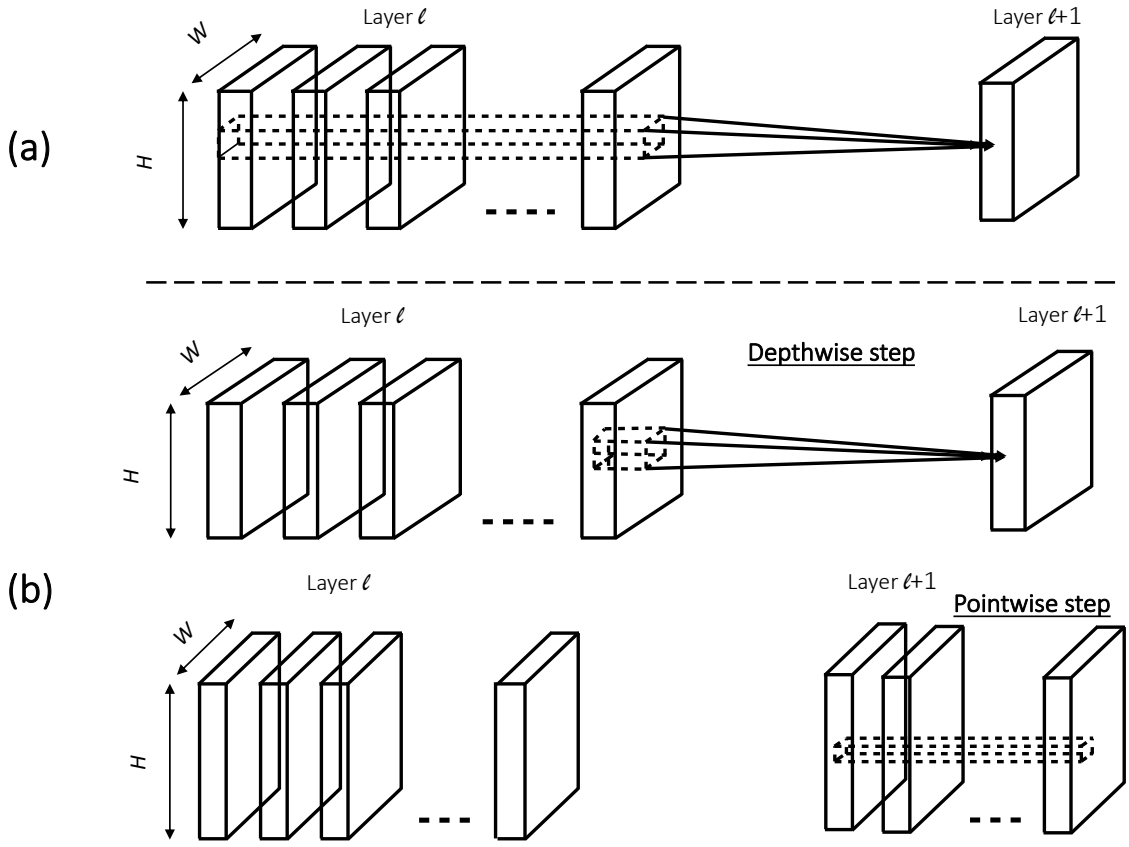


Figure 2.4: (a): Conventional convolutions mix the input channels spatially and channel-wise in a single step (**top row**). **(b)**: In separable convolutions, first each individual input channel is convolved spatially in the depthwise step (depicted in **middle row**). The feature maps are then stacked together and mixed channel-wise via 1×1 convolutions (**bottom row**). The sequential application of the depthwise and pointwise steps is termed a *separable convolution*. H and W give the dimensionality of the feature maps in the figure.

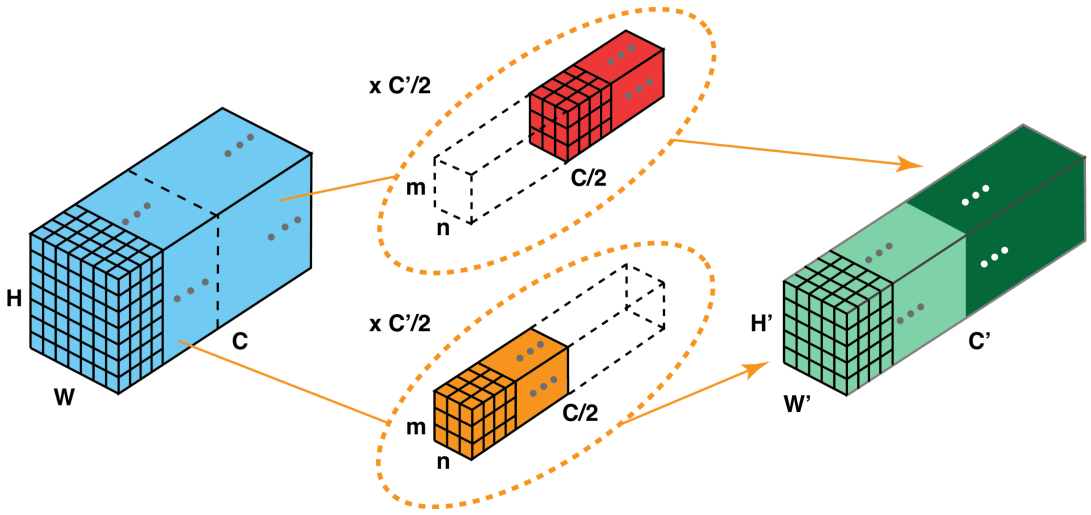


Figure 2.5: An illustration of the grouped convolution operation. In this example, the input channels and kernel depth are split in two groups. Each group undergoes a conventional convolution operation and the resulting activation maps are stacked together to produce the output.

C input channels are first split in several groups, e.g. 2 as is used as an example in Fig. 2.5. Each input channel group is associated with a filter group, which only processes its respective channels. Each filter group has reduced depth compared to conventional convolutions. In this running example the filters contain $\frac{C'}{2}$ kernels and produce this many output channels. The resulting output channels from all groups are stacked together to produce C' feature maps. Grouped convolutions are equivalent to depthwise convolutions when the number of groups is equal to the number of input channels, C. Grouped convolutions provide efficiency advantages both in terms of lower parameterization and learning better representations compared to conventional convolutions. The number of model parameters (compared to nominal convolutions) is reduced proportionally to the number of groups. Also, filter relationships, that is correlations, in adjacent layers are learnt in a more structured way in networks with grouped convolutions. As a result, unneeded filter relationships are no longer parameterized which has a regularization effect on the model and reduces overfitting [102].

Another very commonly used variant of the convolutional layer is the pooling operation. It resembles the convolution in that an $m \times n$ window is overlayed over the input, but instead of element-wise multiplication, a *summary statistic* is calculated. This allows to gradually downsample the spatial dimensionality of the feature maps in the neural network. A very common summary statistic is *max pooling* which reports the maximum value within

Table 2.1: Memory and compute cost between different convolutional layers. H' and W' are the height and width of the output feature map. I disregard the biases for simplicity.

Type	#Parameters	MACs	Relative Cost
Conventional	$m \times n \times C \times C'$	$m \times n \times C \times C' \times H' \times W'$	1
Depth. Sep.	$m \times n \times C + C \times C'$	$(m \times n \times C + C \times C') \times H' \times W'$	$\frac{1}{C'} + \frac{1}{m \times n}$
Grouped	$\frac{m \times n \times C \times C'}{\#groups}$	$\frac{m \times n \times C \times C' \times H' \times W'}{\#groups}$	$\frac{1}{\#groups}$

the window. An important property of pooling layers is invariance to local translation, which can be very useful if we care more about whether some feature is present than exactly where it is, for example in classification tasks.

2.2.3 Residual connections

Theoretical evidence suggest that "deeper is better" when it comes to neural networks [192]. However, optimizing deep networks can be very challenging from a practical perspective. Despite the increased parameterization, simply stacking more layers results in a degradation of both training and testing performance [89]. This counterintuitive behaviour is not yet fully understood. Recent work indicates that the solution to this problem is either increasing the width of the layers, which would not be feasible owing to an exponential dependency, or adding *residual connections* [44].

Fortunately, implementing residual (or skip) connections is simple as illustrated in Fig. 2.6. Assuming a given sequence of layers implements a transformation $\mathcal{F}(\boldsymbol{x})$, a residual connection would modify its computation by aggregating the output with the input, i.e. $\tilde{\mathcal{F}}(\boldsymbol{x}) = \mathcal{F}(\boldsymbol{x}) + \boldsymbol{x}$. This modification proved to be very powerful and enabled the training of very deep neural networks with state-of-the-art results in many application areas.

2.3 Training neural networks

After designing a neural network architecture using fully connected and convolutional operations, we need to learn the parameters $\boldsymbol{\theta}$ of the model in order to produce robust results on some performance measure that is defined with respect to the dataset. To achieve this task, a differentiable *cost function* $J(\boldsymbol{\theta})$ is introduced as the objective function for an optimisation algorithm. The cost function can be written as an average over the

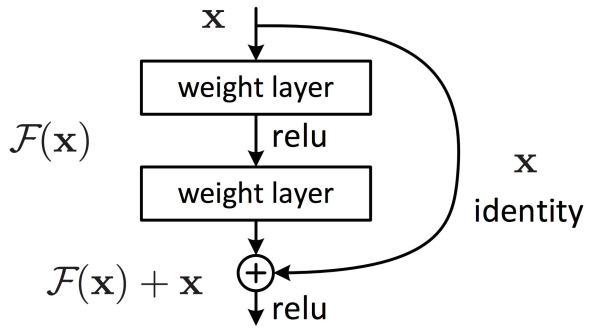


Figure 2.6: A depiction of a residual (or a skip) connection, which shortcuts the input to a neural network transformation, $\mathcal{F}(\boldsymbol{x})$, directly to its output by aggregation. The modified computation is given by $\tilde{\mathcal{F}}(\boldsymbol{x}) = \mathcal{F}(\boldsymbol{x}) + \boldsymbol{x}$. Image taken from [89].

training set:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x}, y_{true}) \sim \hat{p}_{data}} L(NN(\boldsymbol{x}, \boldsymbol{\theta}); y_{true}) \tag{2.6}$$

where L is the per-example loss function, $NN(\boldsymbol{x}; \boldsymbol{\theta})$ is the output from the neural network when the input is $\boldsymbol{x}$ and $\hat{p}_{data}$ is the empirical data distribution. In this example, I assume that y_{true} is a ground truth target that the prediction of the network should match. The cost function from Eq. 2.6 can be rewritten as *empirical risk minimization*:

$$J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} L(NN(\boldsymbol{x^{(i)}}, \boldsymbol{\theta}); y_{true}^{(i)}) \tag{2.7}$$

The parameters which minimize the cost function, $J(\boldsymbol{\theta})$, are considered the best "fit" for the model. What is left to be defined is a suitable loss function, L, depending on the machine learning task.

2.3.1 The maximum likelihood principle

One very common guiding principle to derive a sensical loss function is *maximum likelihood*. Consider that the N training samples $\boldsymbol{X} = \{\boldsymbol{x}^{(i)}, \cdots, \boldsymbol{x}^{(N)}\}$ are drawn independently from an unknown data generating distribution $p_{data}(\boldsymbol{x})$. Then, the neural network transformation $NN(\boldsymbol{x}; \boldsymbol{\theta})$, which represents the probability of the input under the model, $p_{model}(\boldsymbol{x})$, maps an input $\boldsymbol{x}$ to a real number estimating the true probability $p_{data}(\boldsymbol{x})$. Intuitively, the best "fit" for the model parameters maximize the probability of observing the true data under the model:

$$\boldsymbol{\theta}_{ML} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{i=1}^{N} p_{model}(\boldsymbol{x}; \boldsymbol{\theta}) \tag{2.8}$$

After taking the logarithm and dividing by the number of training samples, Eq. 2.8 can be expressed as:

$$\boldsymbol{\theta}_{ML} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \mathbb{E}_{(\boldsymbol{x})\sim\hat{p}_{data}} \log p_{model}(\boldsymbol{x};\boldsymbol{\theta}) \tag{2.9}$$

I present two modelling approaches which employ the maximum likelihood principle to solve classification and regression tasks, and derive their respective loss functions. Maximum likelihood is so commonly used in machine learning modelling because it can be shown to be the best estimator asymptotically as the number of examples $N \to \infty$, in terms of rate of convergence [180].

Regression as maximum likelihood. Since regression is an algorithm which takes a real-valued input $\boldsymbol{x}$ and produces a real-valued output y, the data generating distribution provides both the input and target variables, $p_{data}(\boldsymbol{x}, y_{true})$. In general, the output, y, can be either a scalar or a vector, but in this background section I assume it is a scalar. The key idea is to use the neural network model to produce a conditional distribution $p(y|\boldsymbol{x})$. For regression, the conditional is modelled as a normal distribution with a mean equal to the neural network prediction and variance equal to a fixed constant, i.e. $p(y|\boldsymbol{x}) = \mathcal{N}(NN(\boldsymbol{x},\boldsymbol{\theta}), \sigma^2)$. After substituting with the analytical expression of the log likelihood of the normal distribution, Eq.2.9 can be written as:

$$\boldsymbol{\theta}_{ML} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} ||NN(\boldsymbol{x^{(i)}},\boldsymbol{\theta}) - y^{(i)}_{true}||^2 \tag{2.10}$$

This cost function resembles the one in Eq.2.7. The loss, L, is equal to the mean squared error between the neural network predictions and the ground truth values.

Classification as maximum likelihood. In this setting, the goal is to predict the target label, y, of the input $\boldsymbol{x}$, where the ground truth y_{true} is usually represented as a one-hot encoding over the K input classes. For each input, the output of the neural network, $p(y|\boldsymbol{x})$, can be modelled as a categorical distribution over the classes by passing the activations of the last hidden layer, $\boldsymbol{h}$, through a *softmax function*. The length of the activations vector $\boldsymbol{h}$ should be equal to the number of classes, K. The probability that the input is assigned to class j is:

$$p(y = j|\boldsymbol{x}) = y_j = \frac{exp(h_j)}{\sum_{k=1}^{K} exp(h_k)} \tag{2.11}$$

After transforming Eq.2.9 to a minimization procedure by negating the expression, it can be recognised as the *cross-entropy* between the ground truth targets, y_{true}, and predicted

labels, y. Thus, the cost function in this case becomes:

$$\boldsymbol{\theta}_{ML} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} y_{true,j}^{(i)} \log y_j^{(i)} \tag{2.12}$$

After inspection, the per-example loss function, L, can be identified as the *categorical cross-entropy* loss.

2.3.2 Optimization

Minimizing the cost function, $J(\boldsymbol{\theta})$, is is achieved with an optimization algorithm which iteratively updates the parameters. For this task, the most commonly used property is the gradient:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x}, y_{true}) \sim \hat{p}_{data}} \nabla_{\boldsymbol{\theta}} \log p_{model}(\boldsymbol{x}, y; \boldsymbol{\theta}) \tag{2.13}$$

Practically, for the statistical estimation of the gradient, the expectation is computed by randomly sampling a small number of examples from the dataset, i.e. a batch $\mathbf{b}$, then taking the average over only those examples. Most optimisation algorithms converge faster using such approximate estimates of the gradient. A canonical example is *minibatch stochastic gradient descent* (SGD). A single step of SGD with a batch size of N samples can be represented as:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{1}{N} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{N} L(NN(\boldsymbol{x^{(i)}}, \boldsymbol{\theta}); y_{true}^{(i)}) \tag{2.14}$$

where η is the learning rate. To overcome the noise introduced by the random sampling of minibatches, it is important to decay the learning rate over time for better convergence. To improve learning in high curvature spaces and noisy gradients, modifications to the standard SGD algorithm are required. Here, I briefly present the method of *momentum* [177, 159, 160, 217], although other techniques relying on an *adaptive learning rate* [228, 45] and a combination of the two approaches [109] have been explored. The basic idea of momentum is to accumulate an exponentially decaying moving average of past gradients and continue moving in the same direction. The analogy is derived from physics - the gradient is the force which is moving the model parameters in parameter space, whereas the momentum is the velocity, $\boldsymbol{v}$ (assuming unit mass). The update rule for SGD with momentum is given by:

$$\begin{aligned} \boldsymbol{v}_{t+1} &= \alpha \boldsymbol{v}_t - \eta \frac{1}{N} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{N} L(NN(\boldsymbol{x^{(i)}}, \boldsymbol{\theta}); y_{true}^{(i)}) \\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \boldsymbol{v}_{t+1} \end{aligned} \tag{2.15}$$

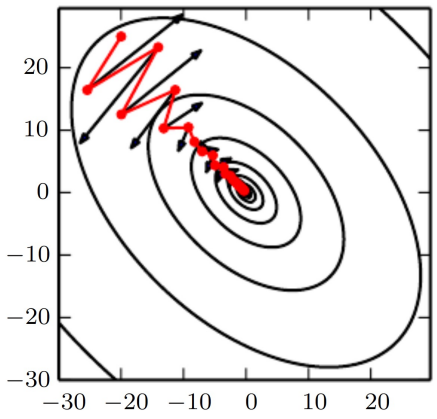


Figure 2.7: The contour lines depict a quadratic loss function. The red line indicates the path of momentum learning and the arrows illustrates the step that gradient descent would take. Momentum can "carry" gradient descent learning over narrow valleys of local minima, whereas vanilla SGD can potentially get stuck and move along the axis of the minima valleys.

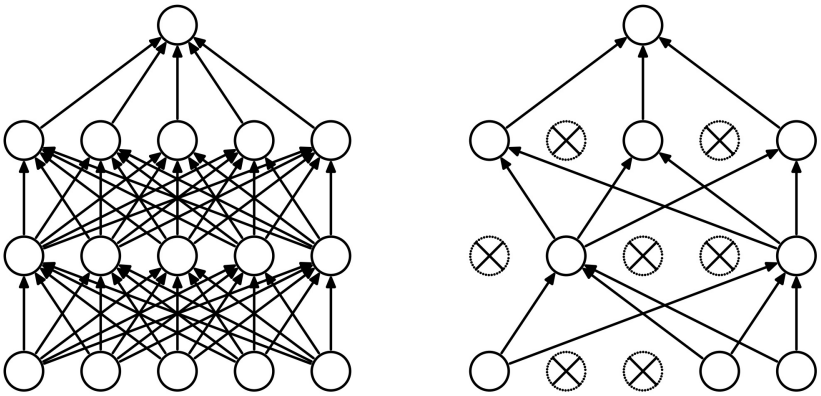
The hyperparameter $\alpha \in [0, 1)$ determines the rate of exponential decay of previous gradients. The higher the α hyperparameter is compared to the learning rate η, the higher the effect of previous gradients. The effect of momentum in SGD is demonstrated in Fig. 2.7.

In practice, before training a neural network model on a dataset, the data is first split in batches, the batches are shuffled, and then a minibatch variant of SGD is run until all batches are exhausted. This marks the completion of a single *epoch*. Usually, we iteratate over the entire dataset for many epochs until learning converges. The gradient of the cost function, $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$, is calculated using the *backpropagation* algorithm [189] by applying the chain rule for partial derivatives, starting from the output layer towards the input.

2.3.3 Regularization

In brief, regularization techniques are modifications to the capacity of a machine learning model intended to reduce the generalization gap but not the training error. In this dissertation, I often use three regularization methods, commonly applied in deep learning, which I describe below.

Weight decay or L_2-regularization describes a preference for the *magnitude* of model parameters. It can be readily implemented as the L_2 norm, and the cost function augmented



(a) Standard Neural Net (b) After applying dropout.

Figure 2.8: Left (a): A standard fully connected neural network. **Right (b):** The resulting thinned network after dropout. The crossed out neurons and their connections have been dropped. Image taken from [210].

to incorporate it:

$$\tilde{J}(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \lambda ||\boldsymbol{\theta}||^2 \tag{2.16}$$

Here, the λ hyperparameter is a penalty term which describes the trade-off between the regularization effect and the original cost function.

Dropout is a simple regularization technique which involves randomly dropping units and their connections with some probability p during training [210]. The key idea is to prevent the neural network from becoming overly reliant on individual neurons. In this way, each neuron is encouraged to learn generally robust transformations which would always be useful regardless of its connectivity. At test time, the network weights are scaled by the retention probability, which approximates Monte-Carlo model averaging of predictions produced by many sampled neural networks.

Batch normalisation is another widely adopted technique which normalizes the inputs to all layers in a neural network during training [103]. Initially, it was thought that the distribution of intermediate layer activations gave rise to "internal covariate shift", hence requiring network weights to constantly adapt to changing input statistics. More recent work demonstrated the effectiveness of batch normalisation came from smoothening the loss landscape [194]. To demonstrate its operation, assume the output activations of a given layer in batch of N examples are $\mathbf{b} = \{\boldsymbol{h}^{(i)}, \cdots, \boldsymbol{h}^{(N)}\}$, and after batch normalisation

we obtain outputs $\tilde{\boldsymbol{h}}^{(i)}$:

$$\boldsymbol{\mu}_{\mathbf{b}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{h}^{(i)} \qquad \boldsymbol{\sigma}^2_{\mathbf{b}} = \frac{1}{N}\sum_{i=1}^{N} (\boldsymbol{h}^{(i)} - \boldsymbol{\mu}_{\mathbf{b}})^2 \tag{2.17}$$

$$\hat{\boldsymbol{h}}^{(i)} = \frac{\boldsymbol{h}^{(i)} - \boldsymbol{\mu}_{\mathbf{b}}}{\sqrt{\boldsymbol{\sigma}^2_{\mathbf{b}} + \epsilon}} \qquad \tilde{\boldsymbol{h}}^{(i)} = \gamma \hat{\boldsymbol{h}}^{(i)} + \beta \tag{2.18}$$

where both γ and β are learnable parameters. They give the batch normalisation layer the additional capacity to learn the best suitable normalisation scheme. At inference, the values of $\boldsymbol{\mu}_{\mathbf{b}}$ and $\boldsymbol{\sigma}^2_{\mathbf{b}}$ are taken across the entire training set.

2.3.4 Hyperparameter selection and validation

Machine learning algorithm design includes making decisions about the values of *hyperparameters*, such as the depth of a network and the kernel size, or the weight decay penalty term. Unlike the model parameters $\boldsymbol{\theta}$, the set of hyperparameters cannot be optimized via SGD on the training set as this will incentivize an infinite increase in the model capacity, for example by adding more layers, such that the training error is minimized. However, this would lead to overfitting. Further, the hyperparameters cannot be tuned on the test set either as this would defeat the purpose of testing the performance of the model on unobserved data. Instead, an additional *validation set* is introduced. Usually, a *grid search* is performed over a hyperparameter range, in which the model is trained with a certain hyperparameter setting, and then the model performance on the validation set is recorded. The highest performing hyperparameter tuning is then chosen and the respective model is tested on the test set.

To gain better statistical certainty about the model's generalization, for example because the dataset is small, the entire train/validation/test procedure can be repeated several times using *k-fold cross-validation*. Here, the entire dataset is partitioned into k folds, using one of them for testing, and the remaining ones for training and validation. This procedure is repeated k times, each time using a different fold for testing. Finally, the test error is estimated as the average across the k trials.

2.4 Structured probabilistic modelling

Discriminative models take an input from a rich highly dimensional distribution and produce a single target variable y, that is they produce a *summary statistic* by learning the conditional distribution $p(y|\boldsymbol{x})$. However, this discards the majority of information

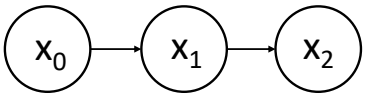


Figure 2.9: An example of a directed graphical model.

in the input. For instance, when classifying an object in an image, the majority of the background can be neglected. In this section, I will present an overview of *structured probabilistic models*, which enable learning more expressive representations from input data. For example, *generative models* learn to estimate the true density, $p(\boldsymbol{x})$, under the data generating distribution. Generative modelling is more challenging than discriminative modelling as it requires a thorough understanding of the input as opposed to parts of it.

Usually, in generative modelling, the input $\boldsymbol{x}$ is referred to as an *observed* variable, as we have direct access to it through the training set. To explain the dependencies between different elements of the input, it is well-established practice to introduce a set of *latent*, or hidden, variables, $\boldsymbol{z}$. This allows for significant *learning efficiency* as we no longer need to model the dependency between any pair of variables x_j and x_i of the input. Instead, this can be captured by the dependency between $\boldsymbol{z}$ and x_i. Another advantage is that the hidden variables $\boldsymbol{z}$ can be used as an alternative, and often lower dimensional, representation of the input $\boldsymbol{x}$. These hidden descriptions are sufficiently expressive to be used as input features for a variety of downstream classification tasks, as well as in representation learning.

Finally, a key benefit of structured probabilistic modelling is that *inductive biases* can be directly introduced in a model by specifying concrete relationships between the hidden and observed variables. For example, if the observed data is a set of edges between nodes in a graph, it is easy to introduce a hidden variable which represents the community assignments of nodes. This is a problem setting encountered in Chapter 5. Since nodes are known to naturally cluster in communities in real-world networks, the generative process already incorporates an inductive bias based on prior knowledge about the world. This regularizes the model in a sensical fashion and utilizes the key takeaway from the *no free lunch theorem*, that is to incorporate problem specifics in model design.

2.4.1 Directed graphical models

Here, I provide a brief overview of *directed graphical models* or Bayesian networks [174], which use graphs to represent interactions between random variables. Each variable is

represented by a node and a directed arrow indicates a conditional dependency between variables. Only these direct interactions need to be explicitly modelled. See Fig. 2.9 for a graphical example.

More formally, a directed graphical model is represented by a directed acyclic graph, the vertices of which define the random variables in the model, and a set of local conditional probability distributions $p(x_i|P_{AG}(x_i))$, where $P_{AG}(x_i)$ gives the parents of x_i in the graph. The probability distribution over $\boldsymbol{x}$ from Fig. 2.9 can be factorized as:

$$p(\boldsymbol{x}) = \prod_i p(x_i|P_{AG}(x_i)) = p(x_0)p(x_1|x_0)p(x_2|x_1) \tag{2.19}$$

2.4.2 Variational inference

After defining a probabilistic model and the conditional dependencies between variables, we need to learn the model parameters on the training set. Employing maximum likelihood learning would require computing the *posterior*, $p(\boldsymbol{z}|\boldsymbol{x})$. For the real-world problems of interest discussed in this thesis, however, exact inference of the posterior is intractable. Consequently, we need to resort to *approximate* methods. Fortunately, there is an established mathematical framework which describes the process of *inference*, that is computing the posterior, as an optimization problem. This allows for utilizing all the optimization tools frequently used in deep learning, such as stochastic gradient descent.

To construct the optimization problem, first we would need to compute the log probability of the observed data, $\log p(\boldsymbol{x};\boldsymbol{\theta})$. Owing to the intractability issue, a lower bound of the data log likelihood, $\mathcal{L}(\boldsymbol{x},\boldsymbol{\theta},\boldsymbol{q})$, is computed instead. This bound is called the evidence lower bound (ELBO):

$$\mathcal{L}(\boldsymbol{x},\boldsymbol{\theta},\boldsymbol{q}) = \log p(\boldsymbol{x};\boldsymbol{\theta}) - KL[q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})] \tag{2.20}$$

where q is an arbitrary distribution over $\boldsymbol{z}$ and will serve as an approximation to the true posterior, and KL is the Kullback-Leibler divergence. Since the KL divergence is always non-negative, the evidence lower bound and the data log likelihood will be equal only if the true posterior $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})$ and its approximation, $q(\boldsymbol{z}|\boldsymbol{x})$, are equal. Hence, inference can be thought of as *finding an approximation, q, which maximizes the lower bound*, and is as close to the true posterior as possible.

2.4.3 Variational autoencoders

There are some key insights from variational autoencoders [110], or VAEs, that I borrow for the purposes of model development in this thesis. The VAE is a directed graphical

model which uses variational inference for learning and can be trained with gradient-based methods. The VAE introduces a differentiable *generative network*, $g(\boldsymbol{z}; \boldsymbol{\theta})$, which transforms samples of latent variables $\boldsymbol{z}$ to samples $\boldsymbol{x}$. The generative network produces the parameters of the distribution over the samples $\boldsymbol{x}$, or $p_{model}(\boldsymbol{x}|\boldsymbol{z})$. For the approximation of the posterior, $q(\boldsymbol{z}|\boldsymbol{x})$, the VAE framework introduces an *inference network*, $f(\boldsymbol{x}; \boldsymbol{\theta})$, which produces the parameters of q. Learning the parameters of the generative and inference neural networks consists of maximizing the ELBO with respect to their parameters. All of the expectations can be approximated with Monte Carlo sampling. Further, the variational autoencoder reformulates the ELBO as:

$$\mathcal{L} = \mathbb{E}_{\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{x})} \log p_{model}(\boldsymbol{x}|\boldsymbol{z}) - KL[q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})] \tag{2.21}$$

As a simplifying assumption, the approximate posterior is often chosen to be a Gaussian, the mean vector and diagonal covariance of which are produced by the inference network. For similar reasons, the prior over the hidden variables $\boldsymbol{z}$, $p(\boldsymbol{z})$, is chosen to be an isotropic Gaussian with the identity matrix as the covariance. More informative designs can also be incorporated, for example data-driven or autoregressive priors [167], removing the assumption about independence between the hidden variables by modelling the full covariance matrix, introducing a probabilistic clustering prior over the latent variables [144] and more. To generate a sample, the VAE first samples $\boldsymbol{z}$ from the prior distribution, then the generator network produces the parameters of $p_{model}(\boldsymbol{x}|\boldsymbol{z})$. Finally, $\boldsymbol{x}$ is sampled from the generative distribution.

Chapter 3

Parameter-efficient deep learning in neuroimaging

I observe that within the neuroimaging research community it is common practice to spatially align all brain images to a common reference. This means a voxel at a given location contains information from exactly the same anatomical feature across all co-registered neuroimages. In addition to facilitating the study of intersubject brain function, this also reduces the need for representational capacity in a neural network model. Based on this observation, I propose to use convolutional operators with lower representational capacity, which also reduce the parameterization of network models aimed at neuroimaging applications. Reducing the computational and memory cost brings the following advantages: first, speeding up medical imaging research turnaround time, and second, democratising research by reducing the need for computational power.

In section 3.1 of this Chapter, I extend separable convolutions [29] to 3D images and make the implementation open source[1]. This enables for deeper skip connection-based [89] neural network architectures for 3D neuroimages while retaining a low parameterization count. I apply this parameter-efficiency design principle to the problem of Mild Cognitive Impairment to Alzheimer's disease prognostication and achieve state-of-the-art results at the time of publication. In addition, I demonstrate that the proposed deep learning model is *interpretable*, in that it recognizes brain areas clinically associated with the development of Alzheimer's disease as highly important for the classification task. Second, in section 3.2 of this Chapter, I employ the same parameter-efficient principle to learning low-dimensional subject embeddings from multi-modal neuroimaging data with a deep autoencoder model. For both approaches, I also quantify the reduction in computation from utilising a parameter-efficient architecture compared to an equivalent model using standard convolutions.

[1]Link to code: `https://github.com/simeon-spasov/MCI`

3.1 Early prediction of Alzheimer's disease

Currently, more than 30 million people have a clinical diagnosis of Alzheimer's disease (AD) worldwide. This number is expected to triple by 2050 due to increased life expectancy and improvements in general health care [8, 53]. Consequently, early diagnosis, which allows for better disease management, can have significant societal benefits. Patients who suffer from Alzheimer's disease experience a deficit in episodic memory, followed by visuo-spatial impairment, spatio-temporal orientation problems, and eventually frank dementia.

AD-related neuropathology can be identified several years before frank AD clinical manifestation [15, 38, 155, 197, 157], and this suggests that the development of AD might be predicted before clinical onset. A common precursor of Alzheimer's is Mild Cognitive Impairment (MCI). MCI is a broad, ill-defined, and highly heterogeneous phenotypic spectrum which causes relatively less noticeable memory deficits than AD. Around $10\% - 15\%$ of MCI patients per year convert to AD [14, 148]. MCI patients who do not develop AD tend to either remain stable, develop other forms of dementia, or even revert to a 'healthy' state, which suggests that MCI is a highly variable and common clinical conundrum which is likely dependent on different pathogenetic mechanisms. This makes the problem of predicting which MCI patients will convert to Alzheimer's (*known as progressive MCI, or pMCI*) and which will not (*stable MCI, or sMCI*) very difficult.

Since Alzheimer's disease follows specific structural neurodegeneration, that is a pattern starting from the subcortical areas of the brain in early disease stages and spread to the cortical mantle in later stages [15], neuroimaging modalities can be used for diagnosis (e.g. PET and MR imaging), as well as blood or cerebrospinal fluid (CSF) biomarkers [143, 5, 85, 224]. Magnetic resonance imaging (MRI)-based biomarkers have attracted interest in diagnosis of AD as well as in predicting MCI to AD conversion because they do not involve the use of ionizing radiation like positron emission tomography (PET), are less expensive than PET, and less invasive than the use of cerebrospinal fluid (CSF) biomarkers. MRI-based indices can also provide multi-modal information regarding the structure and function of the brain within the same scanning session, which is typically advantageous in many clinical settings.

For AD-to-MCI conversion prediction from multi-modal data, I propose a parameter-efficient neural network architecture, aimed at improving the state-of-the-art prognostication. The key advantages of the model are as follows:

1) I use **parameter-sharing** between the tasks of pMCI vs. sMCI and AD vs. Health Controls. This decision is motivated by the similarity of the two conditions (MCI and

Alzheimer's) and is used to boost performance on the more challenging MCI-to-AD conversion task.

2) The proposed neural network model mainly relies on **parameter-efficient 3D separable and grouped convolutions** for feature extraction to maintain a low number of network parameters, hence reduce computational requirements.

I combine *several input streams, such as several structural MRI modalities, as well as demographic, neuropsychological, and APOe4 genetic data.* I implement the proposed method in Tensorflow [1] and Keras with my own implementation of 3D separable convolutions.

3.1.1 Related Work

There has been a growing interest in developing computational tools that are able, by using MRI-based measures, to discriminate AD patients from healthy individuals, or, most importantly, to discriminate the patients with stable MCI from those MCI patients who, in contrast, progress and develop AD.

Different clinical data and imaging modalities have been used so far with a variable rate of success, including for example, PET [28, 156, 157, 248], MRI [56, 154, 157, 229, 248], functional MRI [95] ,cognitive testing ([20, 154], and CSF biomarkers [36, 146, 183, 203]. In this context, Moradi et al. [154] and Tong et al. [229] were amongst the first to: 1) *perform feature selection* to extract informative voxels from MRI volumes via regularized logistic regression, and 2) use the extracted voxels, along with cognitive measures, to produce *support vector machine (SVM)-based predictions*, achieving an area under the Receiver Operating Characteristic (ROC) curve (AUC) between 0.9 and 0.92. Similarly, Hojjati et al. [95] employed baseline resting state functional MRI (rs-fMRI) data to achieve an AUC of 0.95. In their study, the features were engineered by constructing a brain connectivity matrix which is treated as a graph, and the extracted graph measures represented the input of the SVM.

The classification paradigm, shared by most studies, relies on two independent steps. First, independent component analysis (ICA) [198], L1 regularization [154, 229], or morphometry [36, 52], is used to reduce the dimensionality of the data to a smaller set of descriptive factors. Second, these factors are fed into a multivariate pattern classification algorithm. The dimensionality reduction and classification algorithms are two separate mathematical models which involve different assumptions, and *this can result in a loss of relevant information during the classification procedures* [166]. In addition, the most commonly employed classifiers, such as SVM [154, 95, 229] and Gaussian Processes [248], require the use of kernels, or data transformations, which are often chosen from a limited

and pre-specified set. This process maps the data to a new space in which it is presumed to be easier to separate. However, constructing or choosing an application-specific kernel that acts as a reasonable similarity measure for the classification task is not always possible or easy to achieve.

The use of two separate, and methodologically disjoint, analytical pipelines as well as the need to construct ad hoc kernels can be avoided by employing deep learning algorithms. Recently, such deep-learning methods have been applied to AD vs. healthy controls classification problems [98, 133, 173] and pMCI vs sMCI classification tasks [28, 136, 137]. Choi and Jin [28] and Lu et al. [137] have used deep-learning to achieve one of the highest pMCI/sMCI classification performances to-date ($84\% - 82\%$ conversion rate accuracies for these studies respectively). Their predictions were based on a single (albeit highly informative) imaging modality (PET). A more formal summary of the recent studies and classification methods is presented in Table 3.3. Unlike my proposed architecture, none of these deep learning approaches take into account the fact that neuroimages are commonly co-registered, hence lower representational capacity compared to natural images can be used for such tasks.

3.1.2 Dataset and pre-processing

Data used in the preparation of this chapter were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database (adni.loni.usc.edu). The data comprised 435 men and 350 women aged between 55 and 91 years. The majority of subjects identified as white ($> 94\%$) and non-Hispanic (99.98%). All data we used is summarized in Table 3.1. Differences in median age across groups, that is sMCI, pMCI, AD and HC, were tested using Friedman's ANOVA and group $\times$ gender interactions were tested using Fisher's exact test. None of these interactions were statistically significant ($p > 0.05$). For all participants *we used T1-weighted structural MRI* (Magnetization Prepared Rapid Gradient-Echo or MPRAGE), as well as additional clinical data comprising demographics (age, gender, ethnic and racial categories, education), neuropsychological cognitive assessment tests like the dementia rating scale (CDRSB), the Alzheimer's disease assessment scale (ADAS11, ADAS13), episodic memory evaluations in the Rey Auditory Verbal Learning Test (RAVLT), as well as APOe4 genotyping. Some of these clinical features are continuous variables (e.g. age), whereas others are binary, such as the presence of a specific APOe4 gene variant. I specifically selected these MRI and clinical measures to create a classification approach that uses the least invasive, least expensive and more commonly available diagnostic tools in the clinical practice. In other words, the MRI and clinical measures that I included here can be typically collected in non-tertiary or highly specialized medical centers, which maximizes the potential applicability of the proposed

Table 3.1: Demographic, neuropsychological, cognitive assessment as well as APOe4 genotyping data were used in this study. Abbreviations: APOe4 - Apolipoprotein E; CDRSB – Clinical Dementia Rating Sum of Boxes; ADAS – Alzheimer's Disease Assessment Scale; RAVLT – Ray Auditory Verbal Learning Test.

	No. of	Age	Male/	years in	APOe4 level			CDRSB	ADAS11	ADAS13	RAVLT			
	Subjects	(years)	Female	education	1	2	3				immediate	learning	forgetting	% forget
AD	192	75.6 ± 7	103/81	15 ± 2.9	57	86	41	4.4 ± 1.6	18.8 ± 6	29 ± 7.3	23 ± 7	1.7 ± 1.8	4.4 ± 1.9	89.4 ± 21.2
HC	184	74.6 ± 6	92/100	16.3 ± 2.7	144	43	5	0.2 ± 0.9	6 ± 3.8	9.3 ± 5.7	44 ± 10.5	6 ± 2.4	3.7 ± 2.7	33.1 ± 27.7
pMCI	181	73.7 ± 7	108/73	15.9 ± 2.8	61	90	30	2 ± 1	13.5 ± 4.2	21.9 ± 5.5	27.2 ± 6.5	2.9 ± 2.2	4.9 ± 2.1	78.3 ± 27
sMCI	228	72.2 ± 7	132/96	16 ± 2.8	145	67	16	1.2 ± 0.6	8.4 ± 3.3	13.5 ± 5.3	38.5 ± 10	4.75 ± 2.5	4.35 ± 2.6	50 ± 30

method in clinical practice. For example, I did not include PET and Cerebrospinal fluid biomarkers as input measures as these measures are expensive, less diffuse, and potentially more invasive diagnostic tools than the MRI and clinical indices employed here. *All data used in this study is from baseline assessments* (i.e., obtained during the first data collection visit of the subject; no longitudinal follow-up data is used).

It is important to make all T1-weighted (T1w) MRI images conform to a common image template. I use *two different T1 MRI templates* in order to *assess the robustness of our classification methodology to coregistration inaccuracies*. First, I use a custom T1 template specific to this study. The T1 images were co-registered using symmetrical diffeomorphic mapping to each other and averaged iteratively (i.e. the group average was recreated at the end of each iteration). My collaborator N. Toschi performed this procedure. The second template was the Montreal Neurological T1 Template (MNI152 T1 1 mm). After the creation of both templates, all single subject T1w MRI images were nonlinearly registered to both templates. Details on the template registration procedures are provided in [204]. Further, the co-registered structural MRIs are processed and the following image variants are produced as inputs to the multi-modal deep learning architecture:

1 The **co-registered structural MRI images**. No further processing is required.

2 The local **Jacobian Determinant (JD) images** of the nonlinear part of the deformational field taking each image into template space were extracted. Given that AD atrophy results in brain shrinkage, it is expected that the Jacobian Determinant, which is indicative of local image deformation compared to the "average" brain, is useful for classification. The JD maps were used to complement the MRI images as an additional input stream in our model.

3 **Brain area masking**. In order to evaluate how much a priori knowledge about AD

brain pathophsyiology could improve our classification and also how much irrelevant features hamper classification performance, my collaborator Dr. Luca Passamonti defined a set of regions of interest (ROIs) which included only brain areas known to be heavily involved in AD-related atrophy, namely parietal, temporal and frontal lobes in order to perform an inclusion test (see Fig. 3.1). This was based on the Hammers atlas [84].

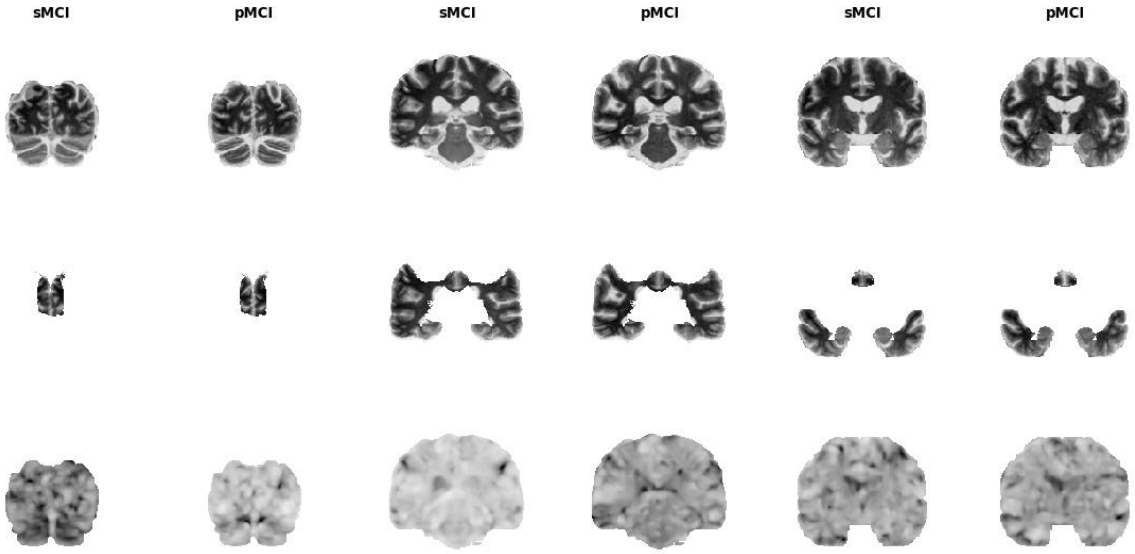


Figure 3.1: Examples of the image inputs we employ in the classification framework for three different image slices. **Upper row**: shows structural MRI images co-registered to a custom common space. **Middle row**: displays atlas-masked MRIs (parietal, temporal and frontal lobes retained), and the **Lower row** shows the Jacobian Determinant images. Data is taken from the Alzheimer's Disease Neuroimaging Initiative (ADNI).

Numerical normalization for the co-registered structural MRI images was performed per sample, i.e. each 3D volume was standardized to 0 mean and unit standard deviation. The reasoning behind this is that brain atrophy could be recognized as an in-sample shift in intensity for a certain area compared to other regions. The normalization applied to the clinical features, i.e. the demographic, neuropsychological, and APOe4 genotyping data, also follows the same feature scaling procedure, where the values of each separate clinical factor are normalized between [0, 1]. On the other hand, the extracted JD images were feature-scaled to have voxel values in the [0; 1] range via subtracting the smallest value in the entire JD image set, and dividing by the difference between the largest and smallest values (also in the entire JD image set). This retains class-wise differences in volumetric changes created when co-registering an image to a template while rescaling the data to a

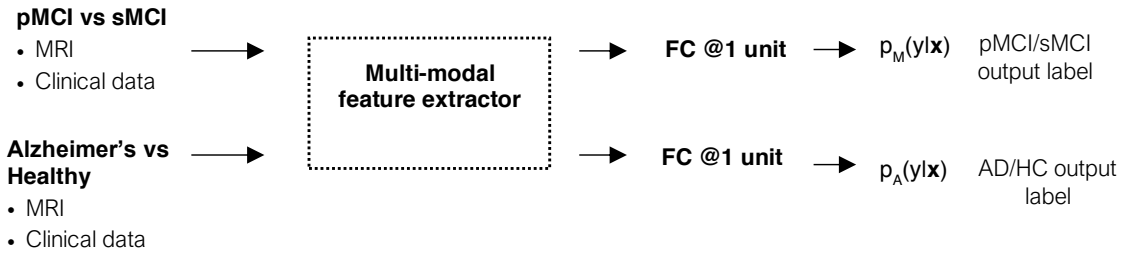


Figure 3.2: An overview of the proposed deep learning model. FC stands for fully connected layers.

global maximum and minimum.

3.1.3 Architecture overview

A high-level overview of the network design is shown in Fig. 3.2. A sub-network, which I refer to as a multi-modal feature extractor, extracts 4-d feature representations from its inputs. This sub-network (with $\boldsymbol{\theta}$ network parameters) is applied on the data from both the pMCI/sMCI and AD/HC discrimination problems, as I assume the *underlying factors of the conditions are similar, hence similar data transformations are likely to be useful*. I then employ two fully connected layers, parametrized by $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$, with sigmoid outputs to approximate the conditional distribution of the labels for the two problems given the inputs ($p_A(y|\boldsymbol{x})$ for the AD vs healthy task and $p_M(y|\boldsymbol{x})$ for the pMCI vs sMCI task). The whole network is trained by minimizing the binary cross-entropy between the observed and estimated labels. The multi-modal feature extractor is represented by a dashed-line rectangle in Fig. 3.2 and Fig. 3.4.

The input data and labels are denoted as pairs $(\boldsymbol{X}, \boldsymbol{Y}) = \{(\boldsymbol{x}_1^A, y_1^A), \cdots, (\boldsymbol{x}_N^A, y_N^A), \cdots, (\boldsymbol{x}_1^M, y_1^M), (\boldsymbol{x}_{N'}^M, y_{N'}^M))\}$, where $\boldsymbol{x}_i^A$ is the i-th observation from the Alzheimer's and healthy subset, and $\boldsymbol{x}_j^M$ j is the j-th observation from the pMCI vs sMCI subset. I refer to the empirical distributions over the AD/HC and MCI subsets as $p_A(\boldsymbol{x}, y)$ and $p_M(\boldsymbol{x}, y)$ respectively. Learning the network parameters can be represented as maximizing the probability of observing the data under the model:

$$\underset{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\psi}}{\operatorname{argmax}} \mathbb{E}_{\boldsymbol{x}^M, y^M \sim p_M(\boldsymbol{x},y)}[\log p_M(y = y^M|\boldsymbol{x}^M)] + \alpha \mathbb{E}_{\boldsymbol{x}^A, y^A \sim p_A(\boldsymbol{x},y)}[\log p_A(y = y^A|\boldsymbol{x}^A)] \quad (3.1)$$

I introduce the α hyperparameter to control the trade-off between the two tasks during learning, and use $\alpha = 0.25$ in all experiments. This is a heuristic choice based on the observation that the AD/HC problem is much easier than the pMCI/sMCI problem and that the model quickly achieves high validation accuracy with this value of α.

3.1.4 Network architecture

The approach proposed in this chapter differs from previous deep learning methods for neuroimaging because it takes into consideration the specific needs of such data compared to natural images. Specifically, that it comes in a specific co-registered format with a lot variance than natural images. Hence, when designing my model, I rely on 1) sharing parameters between two common tasks which share underlying factors, and 2) using parameter-efficient as opposed to conventional convolutions as I argue the representational power of the latter is not needed. Considering the kernel sizes and number of filters in our network architecture, substituting a single conventional convolutional layer with a separable one results in 20 times fewer parameters for that layer. As a result, the network model comprises 557,000 parameters, which is orders of magnitude lower than conventional 3D CNNs and even lower than recent[2] 2D CNNs , such as AlexNet [117] and Xception [29]. I assume that similar descriptive factors and learnt features are useful for both AD/HC and pMCI/sMCI discrimination. Hence, I introduce the auxiliary task of AD/HC classification to increase the number of training samples. While previous methods employ pre-training [173, 98] to reap similar benefits, this requires training the model twice, whereas dual-learning is a single-stage procedure.

Since several different sequences of operations (e.g. convolutions or linear layers) are frequently reused in the network architecture, they are *combined in operational blocks*. All of these blocks are shown in Fig. 3.3. Each block follows a similar pattern. For instance, convolutional blocks, or conv blocks, used to processes the 3D MRI tensor images, comprise a convolutional kernel with linear activations, batch normalization and an exponential linear unit (ELU) transformation with dropout. In order to reduce the resulting spatial dimensions, max pooling is used, with a window of 3 pixels and a stride of 2. The clinical features undergo a series of transformations by fully connected (or FC) blocks. In the FC block, a linear layer is employed but the same regularization precautions and activations as in the conv block are applied. I also implement a separable convolutions block, or sep conv block, which substitutes standard convolutions with separable ones and does not rely on any pooling operations. I provide an implementation of the 3D separable convolution module as a custom Keras layer based on a TensorFlow backend here: https://github.com/simeon-spasov/MCI/blob/master/utils/sepconv3D.py. Fig. 3.4 shows

[2]The methodology of this chapter was largely developed in 2017.

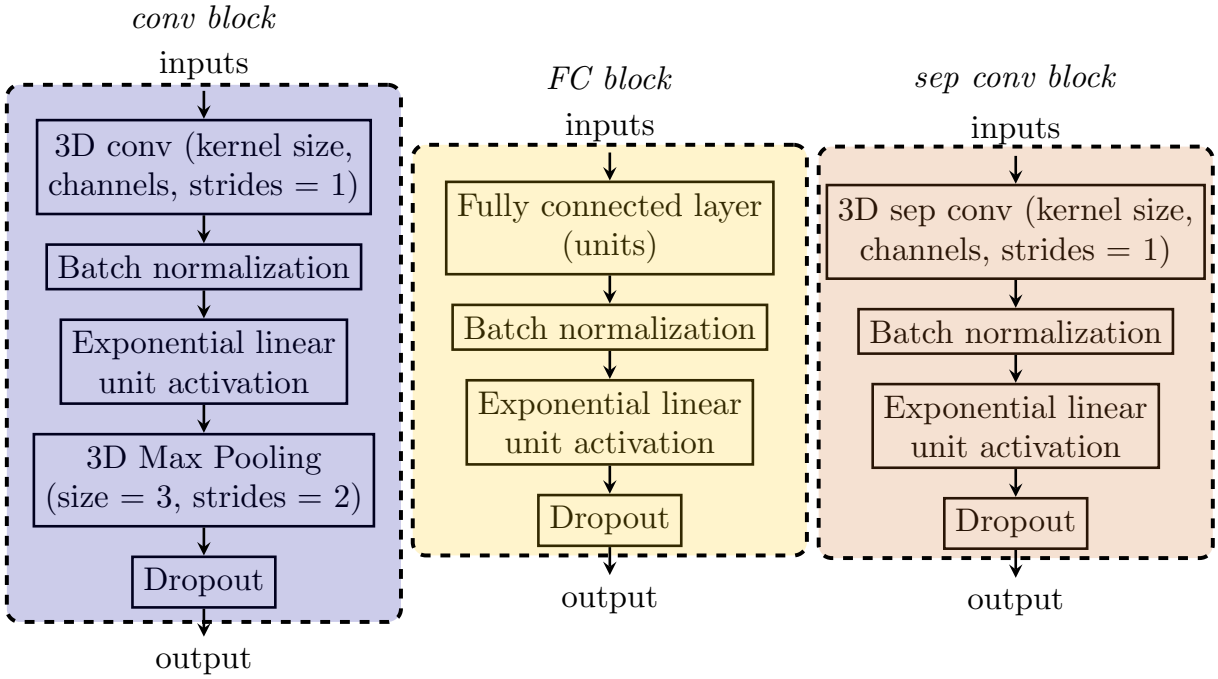


Figure 3.3: Implementation of the operational blocks used in this work. **Left**: convolutional (conv) block, **middle**: fully connected (FC) block, **right**: sep conv block.

the full neural network architecture I use for the AD/HC and pMCI/ sMCI classification problems.

Fig. 3.4 depicts the overall neural network architecture. Firstly, two consecutive convolutional blocks are used to reduce the dimensionality of the input MRI and Jacobian images. The outputs of the second conv block from the MRI and the Jacobian images are concatenated along the channel axis. The majority of the feature extraction is then performed by three sequential separable convolutional blocks. The dimensionality of the activation maps remains the same during this procedure. The output from the last sep conv block is summed element-wise with the activation maps from the second conv block in the add block (also known as a residual connection, introduced in [89] and [29]). It has been shown that residual connections facilitate training as the depth of the neural network increases. I now split the result of the summation along the channel axis in two groups to perform a grouped convolution. The motivation behind opting for grouped convolutions is to further reduce the dimensionality of the activation maps which is not possible by using the fully separable convolutions as outlined in eq. (6) but is more parameter-efficient than utilizing traditional convolutions. At this stage of the image processing pipeline the shape of the activation maps is 1x2x1 with 16 channels after concatenation (8 channels in each group). I flatten the feature maps to a 32-dimensional vector and apply a fully connected block with 10 output units. This 10-dimensional vector forms the final embedding of the MRI and Jacobian images. The clinical features undergo 2 sequential transformations by

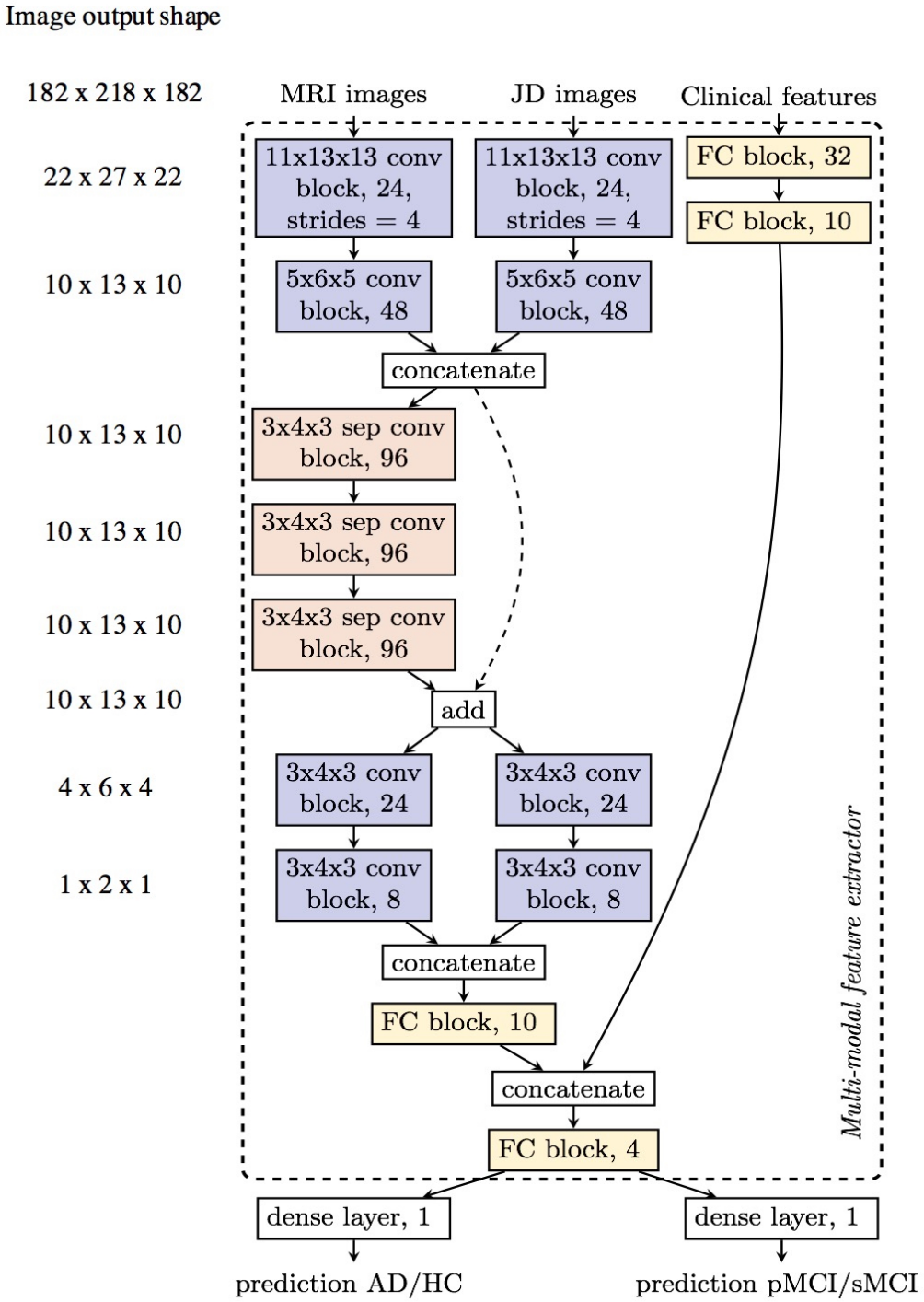


Figure 3.4: Architecture of the neural network designed to take multiple 3D image volumes and clinical variable inputs. The design of the network relies on the operational blocks shown in Fig. 3.3. For conv and sep conv blocks I use the notation: kernel size, (sep) conv block, output channels. The operational blocks are color-coded for the ease of the reader both in this figure and Fig. 3.3. The multi-modal feature extractor sub-network is shown in the dashed rectangle.

fully connected blocks with 32 and 10 units respectively. The clinical features and image embeddings are concatenated and processed by a fully connected block with 4 output units. All of these operations acting on the MRI, Jacobian and clinical feature inputs which ultimately compress the input data in a 4-dimensional vector comprise the Multi-modal feature extractor. In order to obtain a prediction for each of the two tasks (AD/HC and pMCI/sMCI), I pass the 4-d output of the feature extractor sub-network through two dense layers with sigmoid activations and single output units.

3.1.5 Implementation

All experiments were conducted using python version 2.7.12. The neural network was built with the Keras deep learning library using TensorFlow as backend. The 3D separable convolutions are built as a custom layer in TensorFlow. For the experiments in this chapter, I employed a Linux machine and two Nvidia Pascal TITAN X graphics cards with 12 GB RAM each. The model was parallelized across GPUs such that the feature extractor network works on the AD vs HC and MCI-to-AD conversion problems simultaneously to speed up training. Iterating over the whole training set once, i.e. a single epoch, takes about 30s and prediction for a single MCI patient requires milliseconds. Since prediction would not require model parallelization or a lengthy training process, a pre-trained network is practical to be applied on a lower-end GPU (or possibly a CPU) relatively cheaply in a realistic scenario. Across all experiments certain network settings remain unchanged. These include the dropout rate - set at 0.1 for all layers and blocks; the L2 regularization penalty coefficient set at $5 * 10^{-5}$ for all parameters in convolutional and fully connected layers; and the convolutional kernel weight initialization which follows the procedure described by [88]. The objective function loss is minimized using the Adam optimizer by [109] with an exponentially decaying learning rate:

$$lr = 0.001 * 0.3^{epoch/10} \tag{3.2}$$

All other parameters are kept at their default value provided in the original Adam paper [109]. The network hyperparameters were picked because they resulted in sufficiently good performance on the validation set. A training batch size of 6 samples for both the AD and MCI conversion problems is randomly sampled from the dataset when training the network until the dataset is exhausted.

3.1.6 Performance Evaluation

For the evaluation of the classifier, I repeated the sampling strategy to divide the samples in training, validation and test set splits. Since there are 32 samples more in the MCI

dataset (16 for pMCI and 16 for sMCI) as compared to the AD/HC dataset, I used these 32 MCI subjects for testing purposes by randomly sampling 16 subjects from the pMCI and sMCI groups. The validation set comprised roughly 10% of the remaining dataset (36 subjects from MCI and AD/HC respectively) and was also generated by randomly picking in a balanced manner from the progressive and stable MCI groups, and from the healthy and AD patients. Finally, the remaining 340 subjects from both the AD/HC and MCI subsets respectively (i.e. a total of 680 subjects) comprised the training set. No data augmentation procedures were used as they were deemed computationally prohibitive and slowed down training significantly.

The model is trained for 40 epochs and the best performing model with the lowest loss on the validation set is saved and its performance is evaluated on the test set. This procedure is then repeated 10 times with different sampling seeds so as to have different samples in the train/validation/test splits (or folds) and minimize the effect of random variation. The number of subjects in each of the training/validation/testing splits is maintained the same at 680/72/32 subjects respectively. The evaluation metrics used and reported in our results are accuracy (ACC), sensitivity (SEN), specificity (SPE). I also perform receiver operating characteristics (ROC) analysis and compute the AUC across folds. The optimal operating point of the ROC curve was found via Youden's J statistic. All accuracy, sensitivity and specificity results are reported at the optimal operating point of the ROC curve. For the AD vs HC task, I report the validation results as there is only a test set for the pMCI/sMCI classification problem (while the AD/HC task is a helpful auxiliary problem, it is a very easy classification problem which is also not the focus of this work).

3.1.7 Results

Firstly, I consider the classification performance of the network on *four different input biomarker combinations*. The four input combinations are:

1) clinical features and T1w MRI images;

2) clinical features and Jacobian Determinant images;

3) clinical features and atlas-masked T1w MRI images;

4) clinical features, Jacobian Determinant and T1w MRI images.

I performed all of these experiments in custom template space. In order to assess the robustness of the neural network model to MRI structural misalignment, I also performed *three experiments in the MNI152 T1 template space* with three different input

combinations (all input combinations except for 3), i.e. without clinical features and atlas-masked T1w images). Under the assumption that using the custom template will result in higher co-registration accuracy as compared to using the MNI template, the purpose of these experiments is to assess the robustness of the methodology to possible structural misalignment.

Classification performance

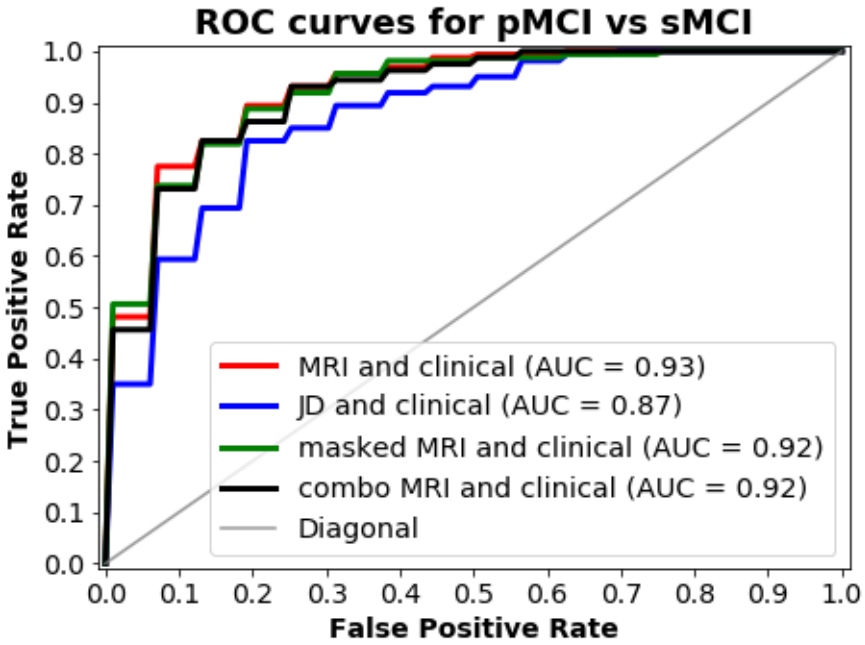


Figure 3.5: ROC curves on the pMCI vs sMCI classification task for four input combinations: MRI images and clinical features; JD images and clinical features; Atlas-masked MRI (or just masked MRI) images and clinical features, and finally MRI and Jacobian Determinant images and clinical features. The MRI data was coregistered to our custom template prior to performing classification. The grey ROC curve at the diagonal was generated by randomly permuting the training labels for the structural MRI and clinical features input combination and predicting using this random classifier.

Results are summarized in Fig. 3.5 and Fig. 3.6 and Table 3.2. The best performance metrics are achieved by including structural MRI along with all clinical data (demographic, neuropsychological, and APOe4 genotyping features). The median AUC across folds for the input combination comprising structural MRI images and clinical features is 0.925 whereas when we remove brain areas not classically associated with AD (i.e. using the atlas-masked images we employ in the inclusion test), the median AUC obtained is 0.922. Comparing these results across folds using a Mann-Whitney U test indicated that *removing brain structures unrelated to the development of AD does not hinder or aid ($P = 0.4$) discrimination in pMCI and sMCI*. The median AUC when using JD images and clinical data was

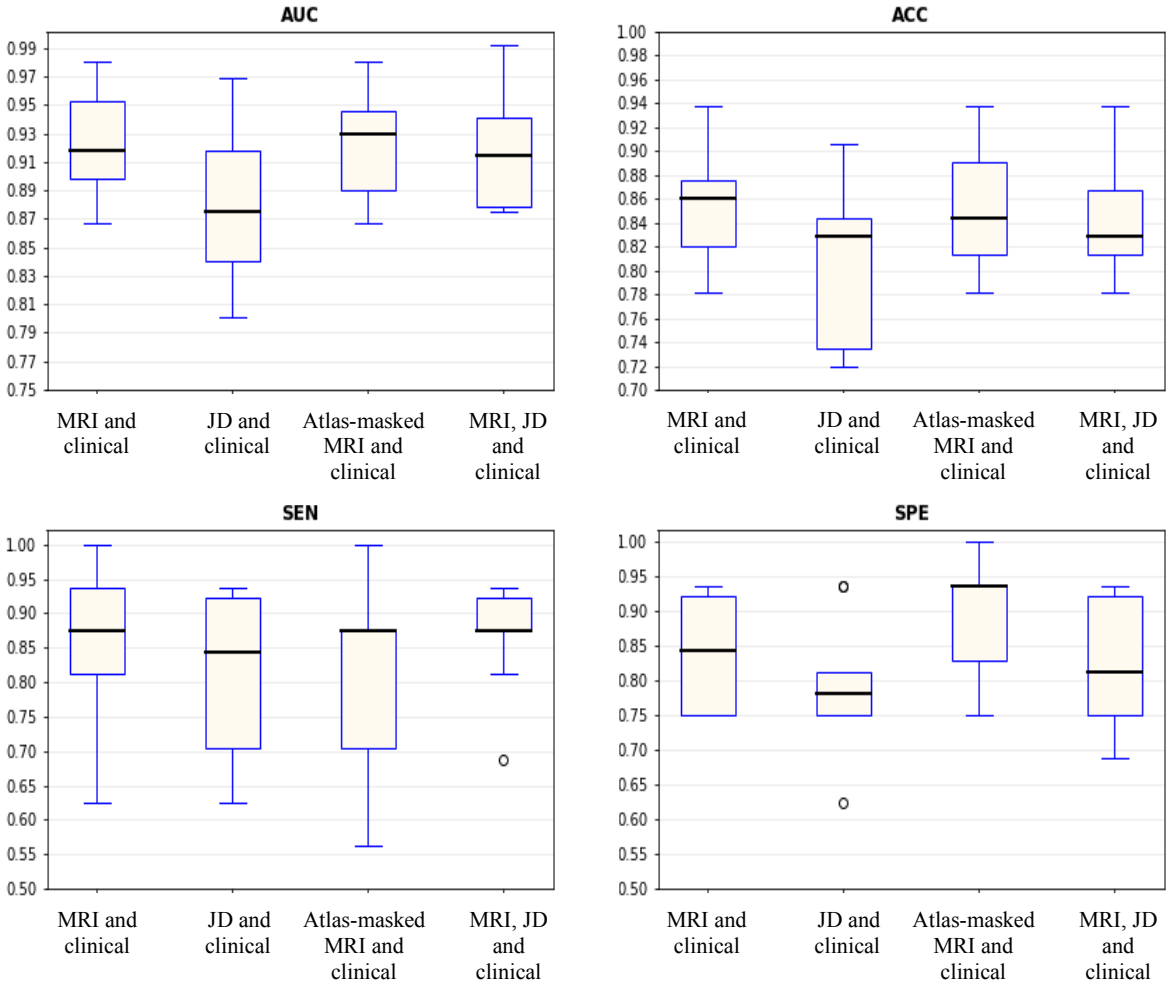


Figure 3.6: Box plots for AUC, accuracy, sensitivity and specificity on the pMCI vs sMCI classification task based on multi-modal integration of clinical features and MRI images (co-registered to our custom template) over 10 separate test folds. The black line in each box represents the median value. The boxes encompass values between the 25^{th} and 75^{th} percentile whereas the tails - the top and bottom quartiles. Outliers are marked with a circle. The performance metrics correspond to the optimal operating point of each classifier.

Table 3.2: A comparison table between the median performance metrics on the pMCI vs sMCI classification task using our neural network model.

Input modalities	Custom template				MNI152 template			
	AUC	ACC	SEN	SPE	AUC	ACC	SEN	SPE
MRI and clinical	0.925	86%	87.5%	84%	0.917	85%	82%	87%
Atlas-masked MRI and clinical	0.922	84%	87.5%	94%	-	-	-	-
JD and clinical	0.874	83%	84%	78%	0.881	82%	82%	81%
MRI and JD and clinical	0.917	83%	87.5%	81%	0.899	83%	77%	88%
structural MRI	0.790	72%	63%	81%	-	-	-	-
clinical data	0.880	81%	83%	81%	-	-	-	-

found to be 0.874 (Mann-Whitney test yielded p-value = 0.041 and 0.046 when compared to the input combinations comprising structural MRI and clinical data, and atlas-masked structural MRI and clinical data results, respectively). Finally, the input combination comprising all types of input streams - T1w images, JD data and clinical features resulted in an AUC of 0.917. Comparing this with the input variants comprising the structural MRI and clinical features, atlas-masked MRI and clinical features, or JD images and clinical features yielded p-values of 0.36, 0.38 and 0.07 respectively (Mann-Whitney-U test).

These results suggest that:

1) Adding **structural MRI** to the clinical features yields statistically significant **higher performance** as opposed to using the JD data as an image input stream.

2) Removing brain areas from structural MRI not classically associated with Alzheimer's disease did not show statistically different classification results compared to the experiments which retained all information. This suggests **the model was not negatively impacted by the inclusion of irrelevant or only partially relevant features**.

In addition to point 2), this experiment corroborates the expectation that areas associated with AD development would possess the highest discriminative power between pMCI and sMCI, and also demonstrates a possible practical avenue for relating subsets of the input feature space to the predicted outcome with deep learning methods. This avenue of research can *boost trust in and/or add interpretability* to deep learning approaches in that they respond to sensical parts of the input space.

The highest median classification accuracy achieved was 86%, which resulted from the experiments with structural MRI and clinical data. The atlas-masked MRI and clinical data variant yielded the second best result with 84% classification accuracy, whereas the JD images and the clinical features gave 83% accuracy. Finally, employing all input features also resulted in an accuracy of 83% . Across the classification results from the four different input combinations the median sensitivity varies between 85% and 87.5%, and the median specificity between 78% and 94% (evaluated at the optimal operating point of each ROC curve across the test folds).

Results from the classification performance on both the custom and the MNI152 template are summarized in Table 3.2. I performed Mann Whitney U tests across folds on the obtained AUCs corresponding to the different input combination pairs (custom template vs MNI template). The obtained p-values are 0.28, 0.42 and 0.24 for the structural MRI and clinical features, Jacobian Determinants and clinical features, and the combined inputs respectively. Consequently, *no statistically significant difference can be found between the performance of the network classifier while operating in the two different normalization spaces* (custom template vs. MNI).

In addition, in order to identify the relative contribution of the structural MRI images compared to the clinical variables, I ran the pMCI vs sMCI performance evaluation procedure using either structural MRI images (not Jacobian determinant images), or clinical features as inputs. Using the clinical features alone resulted in an AUC of 0.88 (average across folds), whereas using only the structural MRI data resulted in an averaged AUC across the folds of 0.79.

Owing to the simpler nature of AD vs HC discrimination, regardless of the input streams and the co-registration template, results are close to 100% on all performance metrics.

Comparison to previous studies

The classification performances of the network model proposed in this chapter are higher than that reported in previous studies except for the work by Hojjati et al. [95] who used resting state fMRI data. At the time of publishing Hojjati et al., ADNI had made publicly available only a limited set of rs-fMRI data (18 pMCI and 62 sMCI subjects) which makes it difficult to predict how their analytical framework would have scaled to larger populations. Furthermore, the study by Hojjati et al. does not explicitly mention the use of a separate test set which limits the generalizability of their findings (results are reported on a validation set instead of a dedicated test set). To my knowledge, the study by Liu

Table 3.3: A comparative table of methodologies on the pMCI vs sMCI classification task using the ADNI dataset. The Methods column includes both the feature selection procedure(s) and the classification method.

Author	Data	AUC	ACC	SEN	SPE	Conversion time	Validation and testing method	Method
Spasov et al. (this chapter) [204]	structural MRI+ cognitive measures+ APOe4+demographics	0.925	86%	87.5%	85%	0-36 months	10-fold cross-validation	CNN
Hojjati et al. (2017) [95]	rs-fMRI	0.95	91.4%	83.24%	90.1%	0-36 months	9-fold cross-validation (report on val set)	graph measures + SVM
Moradi et al. (2015) [154]	structural MRI+ cognitive measures	0.9	82%	87%	74%	0-36 months	10-fold cross-validation	LASSO + SVM
Liu et al. (2017) [131]	structural MRI+FDG-PET cognitive measures + +APOe4+demographics	0.92	84.6%	86.5%	82.4%	0-36 months	holdout	ICA + Cox model
Korolev et al. (2016) [114]	structural MRI+clinical + plasmaproteomic data medications	0.87	80%	83%	76%	0-36 months	10-fold cross-validation	Joint Mutual Information + Kernel learning
Beheshti et al. (2017) [10]	structural MRI	0.75	75%	77%	73%	0-36 months	10-fold cross-validation	Morphometry+ t-test + SVM
Choi and Jin (2018) [28]	PET	0.89	84.2%	81%	87%	0-36 months	holdout	CNN
Tong et al. (2017) [229]	structural MRI+ cognitive measures	0.92	84%	88.7%	76.5%	0-36 months	10-fold cross-validation	Elastic Net + SVM
Lu et al. (2018) [137]	FDG-PET	-	82.5%	81.4%	83%	0-36 months	10-fold cross-validation	NN

Table 3.4: Memory and compute comparison against competing DL methods.

Type	#Parameters	MACs	Memory cost	Compute cost
Spasov et al.	$422k$	0.925 GMACs	1	1
Spasov et al. (conventional convolutions)	$1.384M$	1.96 GMACs	3.3	2.12
Choi and Jin (2018) [28]	53.02M	16.073 GMACs	125	17.37
Lu et al. [137]	15.25M	0.015 GMACs	36.13	0.016

et al. [131] presented comparable performance (at least in some metrics) to my network model, with 84.6% classification accuracy vs 86% for my work. Liu et al. [131], however, also included FDG-PET alongside the structural MRI and other clinical biomarkers that I have employed which might have improved their classification performance. Moradi et al. [154] and Tong et al. [229] both employed very similar methodology to each other and a dataset (structural MRI and cognitive tests) similar to the one I use in this chapter. Their sensitivity metrics are comparable to my model ($87\% - 88\%$ sensitivity), however they achieve lower specificity ($74\% - 76\%$ vs. $85\% - 94\%$ specificity for my model). I also evaluated the classification performance of my deep learning framework either solely on structural MRI inputs or clinical features. In my model, the use of structural MRI data alone resulted in an averaged AUC of 0.79, which is higher than the AUC reported in a recent study employing similar types of datasets (Beheshti et al. [10]).

Memory and computational savings

I compare the memory and computational savings of my architecture against the deep learning models from Table 3.3. A detailed analysis of the parameter count and multiply-accumulate operations of different convolutional layers is provided in section 2.2.2 of the Background and more specifically Table 2.1. Compared to a fully convolutional implementation of my architecture, the depthwise separable convolutions results in 3.3× memory savings and 2.12× computational savings. A comparison against a contemporary DL method applied to 3D medical images [28] shows the drawback of using conventional convolutions - my architecture uses 125× less memory and is 17.37× more computationally efficient. Finally, I also compare against a fully connected layer architecture [137], which has to resort to image pre-processing to extract metabolic measures to produce vectorised inputs for their network. Not only does this defeat the purpose of automated feature extraction, but also results in high overparameterization (36× over my proposed method).

Current state-of-the-art My parameter-efficient architecture was studied in greater detail by Ramon-Julvez et al. [179] who assessed the impact of image co-registration

on performance. They also use structural MRI data and clinical variable inputs to the network, and find that the exact co-registration procedure applied to the MRI images does in fact influence performance by several percentage points. They were able to push the accuracy of my model to almost 90% on pMCI vs sMCI while achieving 95% sensitivity (at specificity $\sim$ 85%). Since then, the most impactful recent developments have not come from architectural changes because the community had hit a performance plateau for the input combination of structural MRI and the clinical variables presented in this chapter. Grueso and Viejo-Sobera [79] provide a thorough summary of all attempts in the past years and observe a main driver of performance improvements has been the aggregation of highly multi-modal data, for example PET and magnetoencephalography [79]. There have only been a few papers that have been able to go near or even surpass 90% classification accuracy. The best results using an SVM have been obtained by Pusil et al. [178], who claim 100% classification accuracy using magnetoencephalography data from 54 subjects. The small sample size combined with the fact that the study collected their own data instead of using an existing data repository (like ADNI) brings into question the generalizability of the model. Another study by Gupta et al. [82] uses a combination of four different biomarkers: fluorodeoxyglucose positron emission tomography (FDG-PET), structural magnetic resonance imaging (MRI), cerebrospinal fluid (CSF) protein levels, and Apolipoprotein-E (APOE) genotype. They achieve 94% classification accuracy - significantly higher than unimodal approaches, my method and its extension in [179], thus showcasing the power of combining highly multimodal data. A paper by Lee et al. [124], which also utilizes structural MRI and clinical variables, achieves 88% predictive accuracy at 87% sensitivity and 89% specificity. Despite being a deep learning method, this paper also has the added benefit of higher interpretability. The researchers first parcellate the brain using a priori anatomical knowledge. Then, they use a neural network to extract regional abnormality representations to make a clinical decision. These regional abnormality representations can provide additional interpretability to their approach by mapping them back to brain space. This provides a higher level of understanding model prediction compared to my approach, which does not do this explicitly, although I demonstrate the network uses clinically relevant parts of the brain to make predictions. Finally, the most conceptually novel approach is proposed by El-Sappagh et al. [47] who aggregate 11 modalities to predict sMCI vs pMCI and provide explanations represented in natural language form to help physicians understand these predictions. The modalities include PET, MRI, cognitive scores, genetic data, demographic data, patient history, CSF data, lab tests, vital signs, neuropsychological data and physical exams. The paper uses a myriad of steps to extract features from each modality before inputting them through a two-layer model with a random forest as a classifier algorithm. For explainability, the authors use the Shapley Additive explanations (SHAP) feature attribution

framework. In addition, they implement 22 explainers based on decision trees and fuzzy rule-based systems to provide complementary justifcations for every decision in random forest. In the end, they achieve sMCI vs pMCI accuracy of 87% and sensitivity of 86%. Although this is on par with my results, it provides significantly higher levels of explainability.

Use in a clinical setting

The proposed method can be used in a clinical setting as part of a two-stage filtering procedure. At first clinical contact, a highly sensitive test is administered to patients to identify the ones at risk to develop Alzheimer's. Ideally, this test would not be invasive and low-cost. For example, a simple classifier which uses clinical variables as input, e.g. demographic data, neuropsychological and cognitive assessment data. The classifier would be tuned to operate at high sensitivity, which would also create a high false positive rate. Then, a very specific test is administered as second stage filtering. For instance, a highly multimodal version of the proposed parameter-efficient architecture utilising genetic, structural MRI, PET, historical patient data. Note that the optimised version of my method [179], depending on the image normalization employed, can produce specificity of $\sim 90\%$. Additional modalities would only boost this performance. A parameter-efficient implementation would also help adoption by reducing inference costs.

3.1.8 Summary

In summary, in this section of Chapter 1, I describe a deep learning-based method for the prediction of MCI-to-AD conversion within 3 years, by combining baseline (i.e., obtained during the first visit) structural MRI, demographic, neuropsychological, and APOe4 genetic data from the ADNI database. I achieved a very high predictive performance with an average AUC of 0.925, prediction accuracy of 86%, sensitivity of 87.5% and specificity of 85%, which was state-of-the-art at the time of publication. I suggest the use of parameter-efficient convolutional operators as I observe co-registered neuroimages of the brain do not exhibit as much variability as natural images. This observation can be seen as an inductive bias for structure of the input data. Hence, less expressive network designs should be sufficient to achieve strong performance while reducing memory and computational costs. I demonstrate my architecture is $3.3\times$ more memory efficient and $2.1\times$ more computationally efficient than an implementation with conventional convolutions, as well as more efficient that other competitive methods. I show evidence that the network uses relevant anatomical features for the task of early Alzheimer's detection that have previously been related with the development of the disease. Finally, [179] optimize the performance of my arhitecture by experimenting with different normalization procedures of the input data. The results they achieve with my method are the current SOTA on the problem sMCI vs pMCI, excluding methods using additional modalities.

3.2 Multi-modal autoencoding of neuroimages

In this section, I apply the same simplifying assumption about the structure of neuroimage data and its lower requirement for representation capacity compared to natural images as in the previous section of this Chapter. The difference is that I study its applicability in an unsupervised setting, that is if an autoencoding architecture can produce high-quality reconstructions from low-dimensional embeddings. Methods for representation learning have been gaining importance over the last years as it has become progressively more evident that the classic diagnostic labels are unable to accurately and reliably describe the complexity and variability of several clinical conditions. This is particularly true for a broad range of neuropsychiatric illnesses such as depression and anxiety disorders or behavioural phenotypes, such as aggression and antisocial personality. To overcome this issue, diagnostic approaches that rely on more nuanced representation, e.g. personalized clinical features, have been proposed. More specifically, patients' heterogeneity can be better described by grouping individuals into novel categories, which are based on clinical data collected from subjects. Such new patients' groupings are likely to be disjoint and possibly become independent from the currently employed disease categories. This paradigm is often termed precision- and (in some cases) personalized medicine, and is an important step towards improving healthcare. To achieve the goal of personalized medicine, it is reasonable to visualize and describe each patient as occupying their own, unique position in a high-dimensional space that depends on specific pathophysiological mechanisms. The position of a subject in this high-dimensional space gives novel opportunities to optimize the diagnosis and treatment to the individual needs. However, the possibility to translate this into the clinical practice can only depend on the *possibility to generate such 'subject embeddings' based on a variety of multi-modal patient data.*

Neuroimaging data carry a wealth of spatiotemporally resolved information about each patient's brain. Still, for a comprehensive data-driven stratification, all relevant pathophysiological mechanisms should be well-represented in the multidimensional data fed into the algorithm generating the subject embeddigs. In this context, the recent appearance of large data repositories with curated multimodal data (e.g. the Parkinson Progression Marker Initiative (PPMI) [142], the Alzheimer Disease Neuroimaging Initiative (ADNI) [158], the UK Biobank initiative [214], the Cam-CAN dataset [223] and the Human Connectome Project (HCP) [230] are providing novel opportunities as well as challenges to design and evaluate unsupervised methods for multimodal encoding of subject data. The data usually comprises multiple whole-brain imaging modalities of $\sim 10^6$ voxels each, often acquired at multiple timepoints and thorough geno/phenotypic characterization (e.g. genetic, biochemical, biohumoral, and neuropsychological markers). This opens up avenues to *robust cross-modality data fusion.* Nevertheless, the computational and

conceptual challenges in designing data reduction architectures able to exploit voxelwise multimodal 3D imaging data (most deep learning frameworks are designed to learn from 2D images) while 1) *retaining realistic computation times* and, crucially, 2) *extract informative embeddings while reducing data dimensionality* by at least a factor 1000, are severe.

In this section of Chapter 1, I propose and evaluate an autoencoder deep learning architecture based on modular blocks comprising efficient 3D separable convolutions. The motivation behind my design is to efficiently extract low dimensional subject embeddings while:

1) **fusing multiple 3D neuroimaging modalities** on a voxelwise level;

2) performing **heavy dimensionality reduction** with minimal information loss;

3) building an architecture able to **efficiently reconstruct brain images**.

As proof of concept, I test the proposed architecture on the well characterized Human Connectome Project (HCP) dataset, where I use multiple modalities to perform experiments.

In this section, *I mainly focus on demonstrating that the proposed autoencoding architecture can produce high-quality reconstructions* from the low-dimensional embeddings. Further, in a manuscript recently sent for revision [43], my collaborators and I show that the latent multimodal embeddings can be clustered into easily separable subject strata. We find that these strata map to extremely different phenotypical information (including organic, neuropsychological, personality variables) which was not included in the embedding creation process. Hence, we demonstrate a possible application of the deep autoencoder for finding novel subject categories. A comprehensive overview of the paradigm is presented in Fig. 3.7.

3.2.1 Data and pre-processing

The population used for the present study consists of 974 subjects drawn from the publicly available Human Connectome Project (HCP) public data repository located at `https://humanconnectome.org/`. The cohort resulted from downselecting the total 1200 subject to those who had valid and downloadable diffusion MRI data at the time of processing.

Neuroimage data and preprocessing

The Human Connectome Project dataset is heavy on diffusion MRI, which relies on the displacement of water molecules due to diffusion in the brain to generate the MRI images. This improves the spatial resolution of diffusion MRI to the micron-scale versus the more

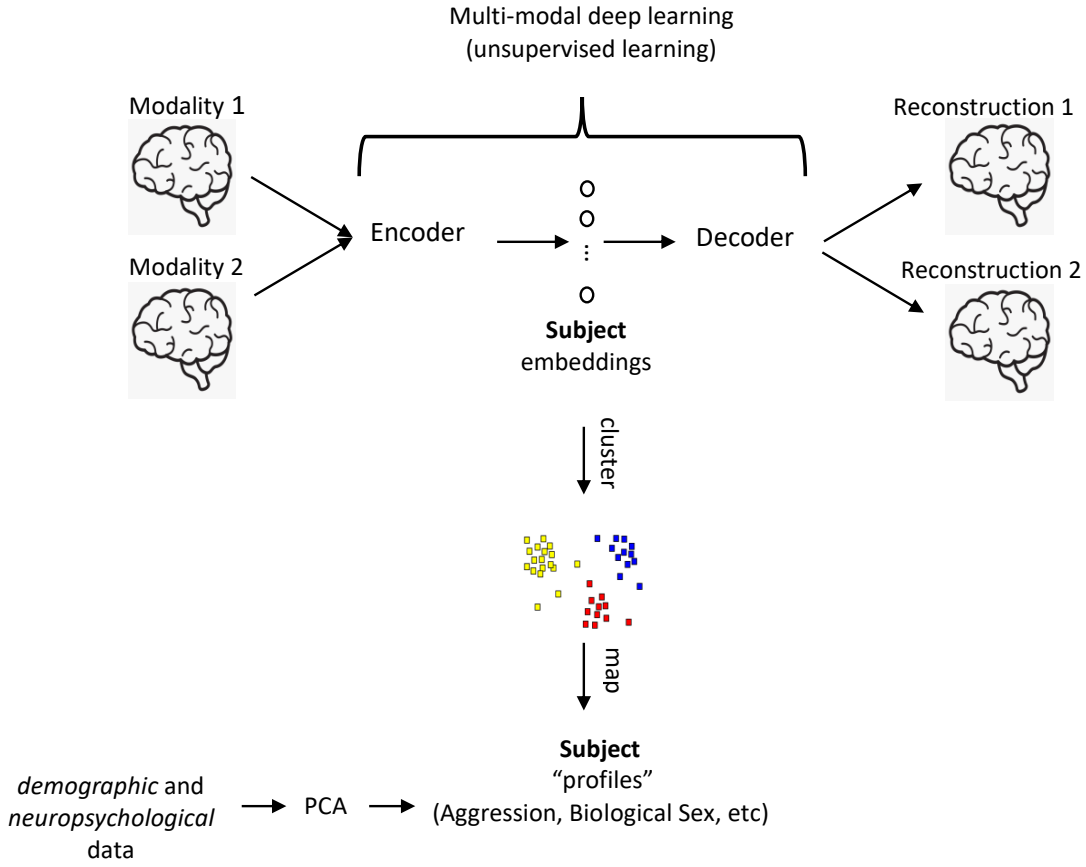


Figure 3.7: A conceptual framework for discovering novel subject categories from multi-modal neuroimage data. First, in this section, I propose a multi-modal deep autoencoder to encode subjects in low-dimensional embeddings. The model down-samples the inputs in jointly learnt embeddings ("subject embeddings" in figure), which summarize information from both modalities. In order to identify novel subtypes, or subject "profiles", it is possible to cluster the subject embeddings and associate them with other phenotypical variables. For example, we can apply a dimensionality reduction technique (Principal Component Analysis or PCA) on these variables and produce interpretable subject "profiles", e.g. biological sex, aggression or neuroticism levels, etc, as shown in the figure. In a way, we give each PCA dimension a human-interpretable category. Then, we can map the clusters produced from the subject embeddings onto these novel and interpretable categories. In this section, I focus on the design of an efficient autoencoding architecture, which generates high quality subject embeddings. More thorough results on the proposed precision medicine paradigm can be found in a pre-print [ref] I produced with collaborators.

conventional structural MRI (e.g T1-weighted) which is at millimeter-scale, and *allows to map axons of white matter* in the brain (i.e. neural tissue microstructure). In this section, I use two different models of diffusion MRI - diffusion tensor MRI (DTI) and neurite orientation dispersion and density imaging (NODDI). From the diffusion images provided by the HCP consortium, a collaborator of mine (Nicola Toschi) extracted *fractional anisotropy maps (FA)*, which are known to be sensitive to microstructural alterations, and *neurite dispersion indices (NDI)*, which are known to be even more specific to the same type of alterations. These image modalities represent scalar values which describe the degree of anisotropy of a diffusion process within each voxel. In addition, I also use *T1-weighted structural MRI*, which adds good contrast between grey and white matter, as well as contrast between some subcortical gray matter nuclei, consequently *complementing well information from diffusion MRI*. The T1-weighted images are non-linearly co-registered to a custom template (similarly to Section 3.1.2 of this Chapter). Then, the *local Jacobian determinant (JAC)* of the non-linear part of the last-stage deformation field which takes each T1-image into template space is extracted. The JAC image quantifies the amount of local volume variation (contraction/expansion) computed when matching the single subject image to the template and is generated in the template space. The same co-registration warp fields were applied to all diffusion derived maps (FA and NDI), resulting in all images (T1, FA, NDI, JAC) in the same resolution space (1.25mm) with high intra-modality and inter-subject anatomical correspondence and downsampled by a factor 2 to yield final volumes of size 128x128x96. Finally, all images were normalized into the 0-1 range by using min-max normalization (0.1 and 99.9 percentile) across the whole population. To summarize, I use the following imaging modalities:

1) T1-weighted (T1w) structural MRI;

2) Jacobian Determinant (JD) images;

3) Fractional anisotropy (FA) maps;

4) Neurite dispersion indices (NDI).

3.2.2 Methods

3.2.3 Multi-modal deep learning architecture

The architecture designed for this study can be viewed as a deep learning, unsupervised autoencoder, composed of a downsampling stage (encoder) and an upsampling stage (decoder) (See Fig 3.8). Several details are inspired from the supervised dual learning framework designed to process 3D medical images proposed in section 3.1 of this thesis. Here, on top of adopting an encode-decode design, in order to improve granularity, I

incorporate skip connections between operational blocks at both stages [185]. The key aspects of my design choices can be summarized as:

1) **Parameter efficiency**: in addition to conventional convolutional layers, I employed alternating sequences of depth-wise and pointwise convolutions, also known as depth-wise separable convolutions [29, 99]. This results in a lightweight neural network architecture with high parameter efficiency and fewer compute operations.

2) **Multi-modality**: it is well established that combining various medical imaging modalities can greatly enhance diagnostic and prognostic performance. I incorporate this data fusion aspect by employing a separate encoder and decoder for each of the modalities, which then all concur to generate a common embedding. The embedded representation is therefore learned (and contains information) from all modalities and can also be used for downstream learning tasks.

3) **U-Net skip connections**: The encoders in the architecture reduce image dimensionality by consecutively pooling the output features of previous layers. This loss of resolution poses a difficulty when learning to reconstruct high-fidelity images with good granularity. U-Net [185] overcomes this challenge, by supplementing the encoder-decoder architecture with contracting paths (or skip connections) between layers in the downsampling and layers in the upsampling stages. In this way the localized high-resolution information is transferred directly to the reconstruction phase.

3.2.4 Architecture details

Figure 3.8 depicts the network architecture of the deep learning autoencoder for medical image embedding and reconstruction (two modalities in the diagram). All modalities have a single channel and voxel values are scaled in the [0, 1] range prior to processing. All training was performed with two modalities (M1 and M2, see Section 3.2.7 for possible combinations and real examples), where each input was a 3D image of size 128x128x96, resulting in approximately 12.2 M data points per modality. The joint multimodal embedding vector is 1536-dimensional (hence representing an approximately 16000-fold data reduction in the encode step).

Both the encode and decode stages are implemented as a sequence of **operational blocks termed conv, mid-flow, "down"conv and "up"conv**. The *parameter-efficient 3D separable convolutions are integrated within the mid-flow block and comprise the majority of convolutional procedures* in the network. At each downsampling step I apply a

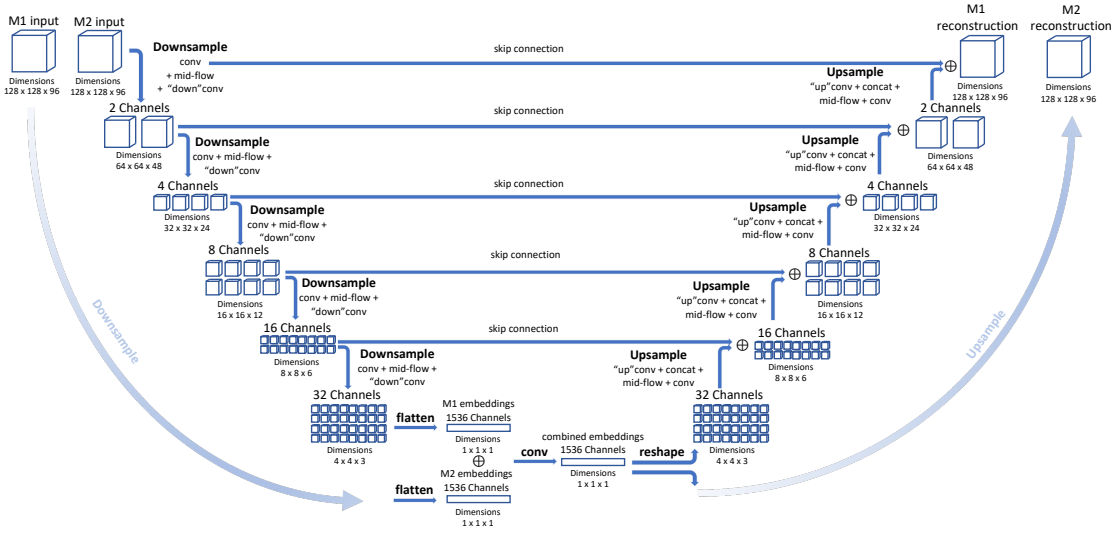


Figure 3.8: The proposed multi-modal architecture for unsupervised learning. The network simultaneously takes multiple modalities (two in this diagram, M1 and M2) and reconstructs both original inputs from a jointly learnt low dimensional embedding, denoted as "combined embeddings" in the figure. All modalities are processed in the downsampling and upsampling stages by successive applications of ad-hoc operational blocks termed conv, mid-flow, "down"conv and "up"conv. The dimensionality of the inputs is reduced via downsampling, and their representations are merged via concatenation ($\bigoplus$ symbol) and convolution. Then, each upsampling stage can be viewed as a mirror image of a downsampling stage at a certain depth. I concatenate corresponding feature maps between the upsampling and downsampling stages, and apply a series of "up" conv, mid-flow and conv block operations to recover the original dimensionality of the inputs. Network parameters are learnt by minimizing the binary cross-entropy between the reconstructions and input images.

sequence of operations comprising "conv", mid-flow and "down"conv, and during upsampling – "up"conv, concat (concatenation needed for skip connections), mid-flow and conv. The inner working of these operational blocks is depicted in Fig. 3.9. In the downsampling stage, I consecutively reduce the dimensionality of the inputs in 5 steps. At each step each dimension of the 3D images is halved, and the number of convolutional filters is doubled. This process is undone during upsampling by upscaling the embeddings by the same factor of 2 at each step.

The downsampling and upsampling streams process both modalities in parallel. I only combine the low-dimensional representations of each input by first flattening them separately, then concatenating (denoted by $\bigoplus$ symbol in Fig. 3.8) and convolving them. To illustrate how skip connections are integrated into the network, consider the output

feature maps with dimensions 32x32x24 and 4 channels in the upsampling stage. First, I "up"conv these activations which scales each dimension twice to 64x64x48. Then, I concatenate these intermediate representations with feature maps produced after a single step of downsampling, which have the same dimensionality, as denoted by the directed arrow labelled "skip connection" in Fig. 3.8. I then mix the combined representations with a mid-flow and a conv block to produce the upsampled feature maps one level higher in the Upsample stream (dimensions 64x64x48 with 2 channels).

3.2.5 Operational blocks

In both the encoding and decoding stages, I employ a series of operational blocks, similar to the ones proposed in Section 3.1 of this chapter, comprising conv, "up"conv, "down"conv, mid-flow (see Fig. 3.9 for a schematic). The most foundational block in the neural network model is the (sep)conv operational block. It has two variants – one working with a standard 3D convolutional operator (conv) and another utilizing 3D separable convolutions (sep conv). In both variants the inputs are first convolved, then batch normalization [103] and an exponential linear unit activation are sequentially applied. I also allow for dropout [210] to be used as the last layer in the (sep)conv block. Both "down"conv and "up"conv are simple extensions of the conv block.

For **"down"conv** I append 3D Max Pooling (size=3, stride=1) to reduce the dimensionality of feature maps in the downsampling stream after conv block processing. On the other hand, in **"up"conv**, I prepend 3D upsampling (size doubles in each dimension) to the conv block. The majority of convolutional processing occurs in an operational block termed **mid-flow**, which has two parallel lines of processing. On one hand, three consecutive sep convs are applied to the inputs. On the other, I introduce a skip connection which does not transform the inputs and allows them to propagate unchanged to the output, which has been proposed to facilitate training of deeper neural networks [89]. Finally, I combine the parallel lines of processing in mid-flow by adding them and returning a single output feature map. *Substituting standard convolutions with separable ones results in a 20-fold decrease in parameter utilization in a single mid-flow block.*

3.2.6 Training and evaluation

The model was implemented in Keras, with a TensorFlow backend and trained on a Nvidia TITAN V GPU. The training was performed over 200 epochs with a batch size of 1. I set the dropout rate at 0.1 for all (sep)conv operational blocks and the L2 regularization coefficient at $5 * 10^{-5}$ for all model parameters. I minimize the binary cross-entropy voxelwise between the reconstructions and the input images using the Adam optimizer

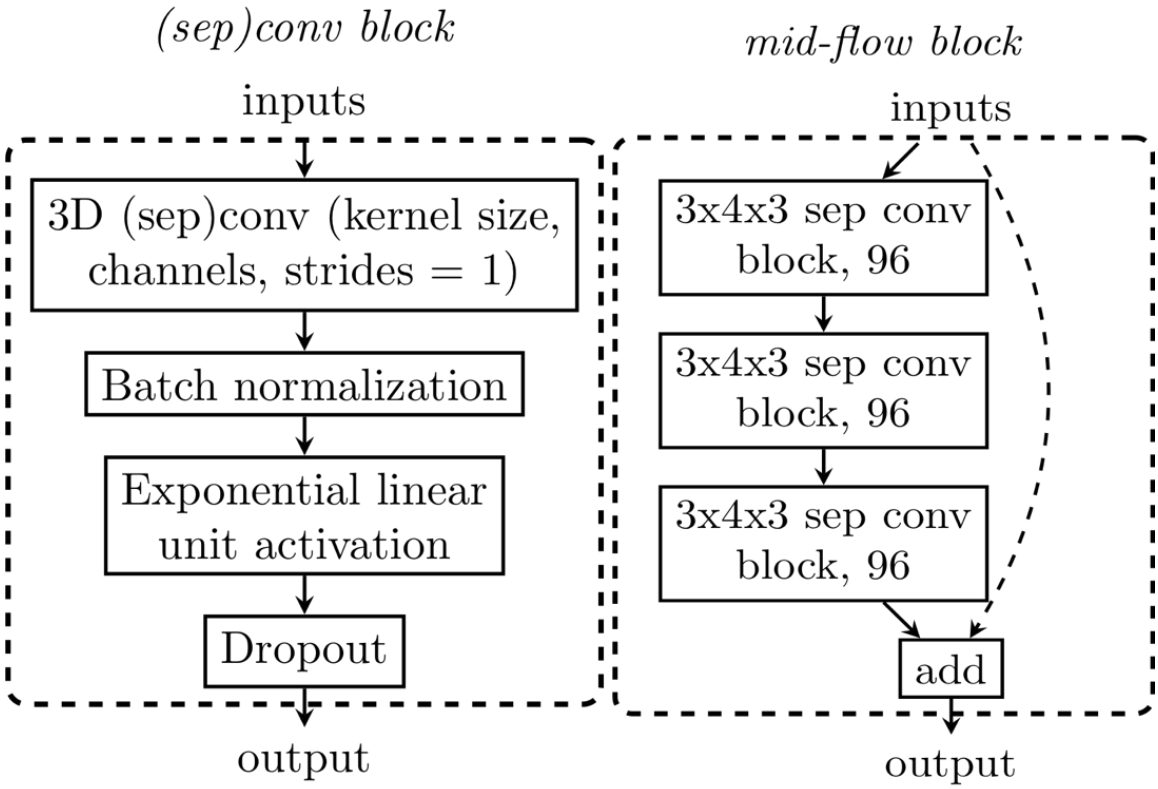


Figure 3.9: Inner workings of the fundamental blocks used in the neural network diagram from Fig. 3.8. **Left:** Since the conv, sep conv, "down"conv and "up"conv blocks are very similar in structure, I use the diagram of the *(sep)conv block* to describe the other derived operational blocks. For example, in "down"conv I apply 3D Max Pooling (size=3, stride=1) at the end of processing, whereas in "up"conv, I prepend 3D Upsampling (size doubles in each dimension); sep conv denotes using separable instead of conventional convolutions. **Right:** The *mid-flow block* is based on a series of three sep conv blocks and a skip connection adding the original input and the output of the final separable convolution. All convolutional kernel sizes in our work are of size 5x6x5 with stride=1. The number of channels can be inferred from Fig. 3.8

[109] using its default settings with the learning rate scheduler given in Eq. 3.2. In order to evaluate the performance of the proposed encoder-decoder architecture, I implemented a 10-fold cross validation procedure using two modalities from the data set of approximately 1000 subjects using 9 folds to train and 1 to test at each run. The following metrics were employed for each fold to assess quality of the reconstructed images, and hence the overall reconstruction performance.

1) **Mean Squared Error (MSE)**: The MSE between each reconstructed and original pair was computed across whole voxelwise images.

2) **"Normalized difference" (NormDiff)**, defined as:

$$NormDiff = \frac{(reconstructed - real)}{(reconstructed + real)} \tag{3.3}$$

This metric is bounded between -1 and 1, and is intended to quantify asymmetry between a real and a reconstructed value. The advantage of this is to eliminate dependency on the underlying image intensity and to focus on relative errors. By calculating this metric voxelwise, I obtained normalized difference images which are then collapsed through the median operator to obtain one value for each real-reconstructed pair

3) **Contrast-to-noise ratio (CNR)** - an important metric in diagnostic imaging. I adopted a simple definition of CNR based on two randomly selected regions of interest (ROI), namely

$$CNR = \frac{(mean(\text{ROI1}) - mean(\text{ROI2}))}{std(\text{ROI1} + \text{ROI2})} \tag{3.4}$$

I then divided each image in regular ROIs of dimension 4x4x3, and randomly sampled 1000 ROI pairs while avoiding sampling from the background. This yielded a 1000-sample estimate of the CNR distribution of each image, which was collapsed through the median operator to obtain one value for each real image and one value for each corresponding reconstructed image. Finally, the reconstructed-real difference in estimated median CNR was evaluated using the normalized difference as defined above.

3.2.7 Results

Visualization. I performed experiments using three pairs of modalities i.e. JAC-FA, JAC-NDI, T1-NDI (Figure 3.10). These choices of pairs were made in order to generate pairs of modalities which, to some degree, complementarily included white matter (WM) and

grey matter (GM) information, that is I combined modalities derived from diffusion *and* T1-weighted MRI as input pairs to the multi-modal autoencoder. The full reconstructed images were inspected by an expert neurologist (Dr. Luca Passamonti) who confirmed the presence of key anatomical details.

Crossvalidation results. In this section, I present the cross-validation results for one exemplary pair of modalities, namely JAC and NDI. This choice was made based on the facts that 1) the Jacobian determinant of a transformation which brings a T1w image into standard space has been seen to be a superior indicator of grey matter density changes as opposed to the T1w image intensities alone; 2) the NDI has shown promise and applicability in a vast number of neuroscience studies (along with a greater sensitivity and specificity in detecting microstructural alterations). Figure 3.11 depicts the fold-wise cross-validation results. MSE was bounded between 0 and 0.02 in all folds, except for fold six where it reached a median value (across all subjects in the fold of approximately 0.038). Considering that both JD and NDI metrics are bounded by definition between -1,1 and 0,1 (respectively) and commonly assume absolute values up to 0.7, these MSE values are around 1/200th of the original intensities.

Memory and computational savings

Because of the complexity of the DL model, it is difficult to calculate the exact number of parameters and MACs of the proposed architecture versus a conventional convolution implementation. Instead, I choose to compare the relative cost savings during a single *Downsample* step as depicted in Fig. 3.8. The Upsampling steps are symmetric and follow similar processing, which means similar considerations apply. A Downsample step comprises a conv, a mid-flow and a "down"conv operation. As a simple approximation, for a conventional convolution implementation this can be seen as a sequence of 5 convolutions. Using the notation from Table 2.1, assuming a relative cost for each convolutional step of 1, the total relative cost is 5 for the conventional case. Substituting 3 of these convolutions for depthwise separable ones for our parameter-efficient implementation, results in an overall relative cost of $2 + 3 \times (\frac{1}{C'} + \frac{1}{m \times n \times k})$, where $m \times n \times k$ is the kernel size, which is fixed at $3 \times 4 \times 3$ for the model, and C' is the number of channels. Since C' varies between 2 to 32, the total relative cost for a single Downsampling step is $2.18 - 3.5$. This is 1.4 to 2.4 times lower than the conventional convolution implementation. Note that this efficiency gain holds for the parameters as well as for the MACs, in this particular example.

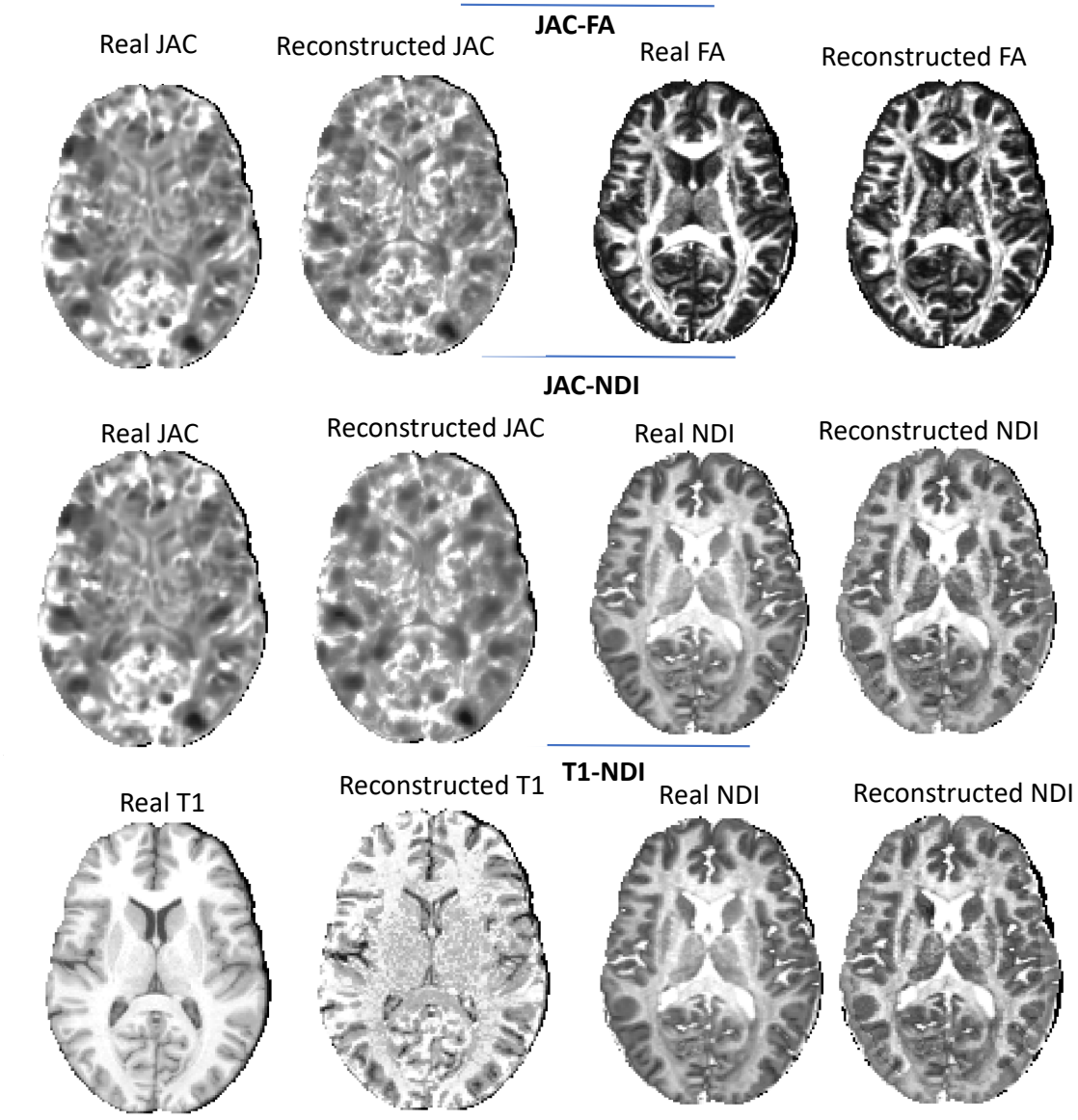


Figure 3.10: Example of pairs reconstructions for JAC-NDI, JAC-FA and T1-NDI modalities, using our multimodal architecture. Color scales are arbitrary but equal for each real-reconstructed pair

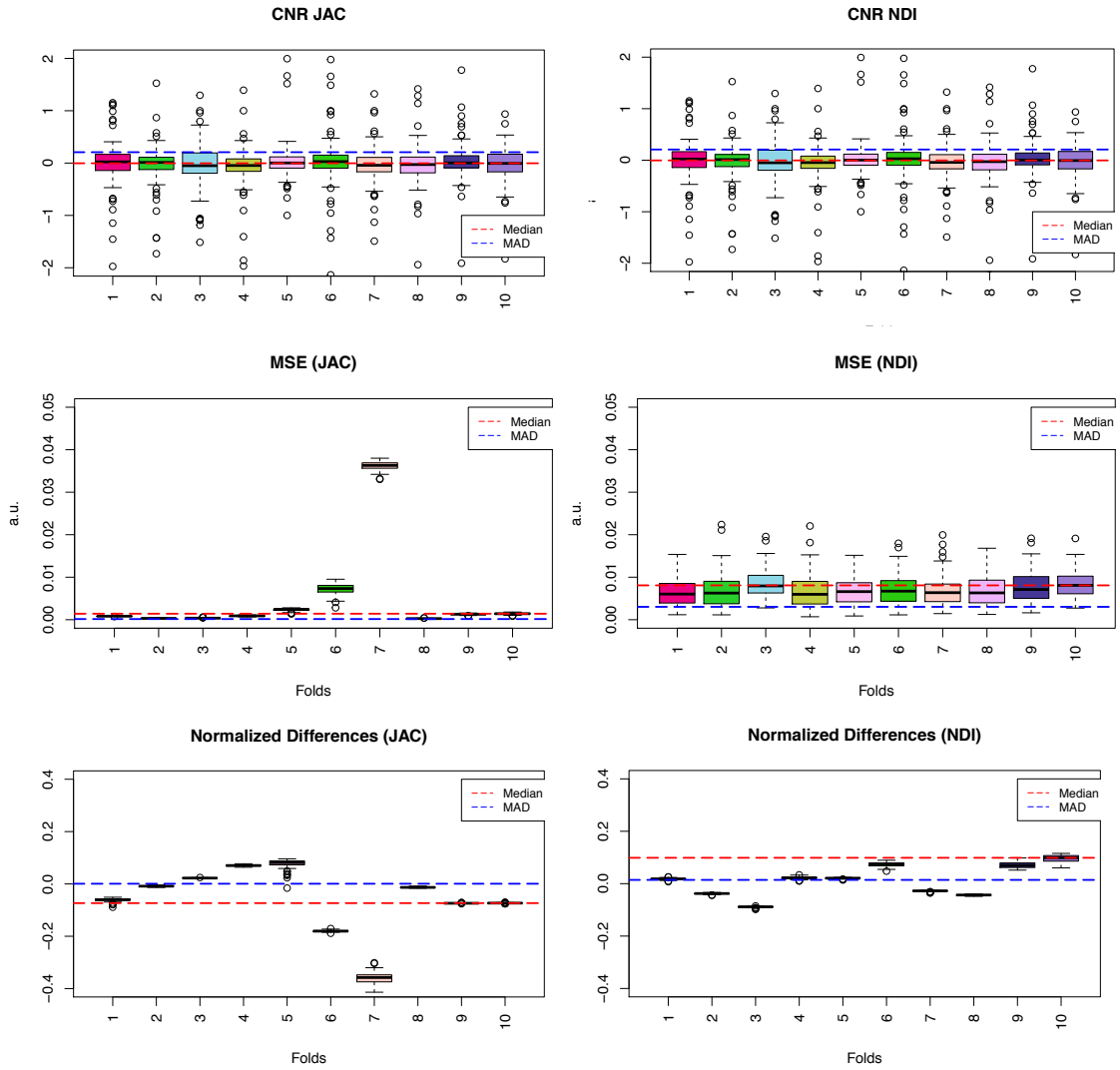


Figure 3.11: In the figure we present the values of CNR, MSE and Normalized Differences for each of the 10 folds when the cross-validation procedure is performed. The boxplots represent the distribution of the values, where the median and the interquantile rages of the distribution are plotted. We also report the values of the Median (with the dotted red line) and of the Median Absolute Deviation MAD (with the blue dotted line) across all of the folds. The x axis represents the folds (from 1 to 10) and the y axis the values of the respective plotted indicator.

3.2.8 Summary

In this section, I propose a deep learning architecture able to compress the extremely large amounts of information present in multiple neuroimaging modalities. I use an autoencoding mechanism based on efficient 3D separable convolutions, which makes the training process less computationally prohibitive in terms of training time and GPU memory utilization. In the experimental section, I assess the reconstruction quality of the autoencoder to ensure that the low dimensional embeddings 1) succeed in encoding high quality information; 2) are able to regenerate the two modalities separately in spite of the complete information fusion implemented in the last encode step. In spite of the extremely large dimensionality reduction, I obtained excellent reconstruction quality in MSE, normalized differences and also CNR, an additional metric commonly used in evaluating the usefulness of medical images. Generating high-quality subject embeddings from multi-modal medical data is an important step towards personalized medicine. In a recent manuscript submission [43], my collaborators and I cluster the produced subject embeddings, associate them with phenotypic/behavioural variables (e.g. age, aggression, etc) and demonstrate the clustering is not random.

Chapter 4

On-device neural network training

The performance of convolutional neural networks (CNNs) in a variety of computer vision tasks, such as image classification, object detection and segmentation, has improved significantly and has surpassed other traditional methods. In recent years, network architectures have become deeper and more complex, yielding highly overparametrized models with a higher memory footprint and many floating-point operations. These requirements have restricted the application of CNNs on resource-constrained devices. Therefore, many techniques, such as pruning [86], which eliminate unnecesary parameters at run-time have been studied. These current *pruning methods*, however, *have solely focused on making inference more cost-efficient.*

On the other hand, little has been done to facilitate training of neural networks on resource-constrained devices. In this chapter, I dynamically identify a high-performing subnetwork within an overparameterized neural network model while simultaneously learning the parameters. I describe a set of modelling assumptions about network architecture, and more specifically parameter independence. This allows me to formulate the problem in the context of combinatorial multi-armed bandits. I propose a reinforcement learning approach based on the combinatorial upper confidence bound algorithm to select a combination of neurons/convolutional channels, which does not introduce any additional parameterization to solve it.

Executing deep learning models on the edge, that is locally on an end consumer product without data exchange with a remote server, provides certain advantages, e.g. enhanced privacy and security. For this reason, on-device AI engines have already been pioneered, and methods for efficient training as well as inference are becoming more relevant.

4.1 Overview

Figure 4.1 provides an overview of the methodology proposed in this chapter. In summary, I draw inspiration from dynamic neural networks [132] and selectively activate and execute a subset of all convolutional channels at each training step. The idea is that *we need to execute, hence load, only the parameters of the activated convolutional channels, consequently reducing the demand for high-memory on-device GPUs.*

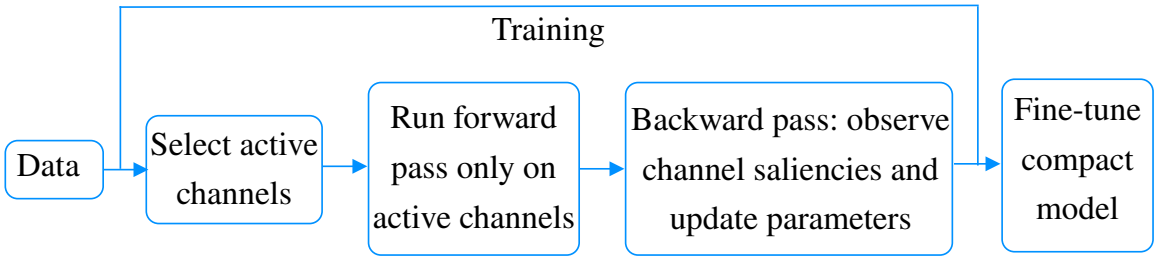


Figure 4.1: Flowchart of the dynamic channel execution framework for efficient training.

In order to be able to select a subset of convolutional channels to execute at each training step, *I introduce a notion of saliency, or weight importance.* For this purpose, a saliency criterion proposed by [153] is used to track the relatively contribution of executed channels to the network performance. Similarly to [257], in this work I assume that a salient channel ought to possess discriminative power regardless of its position in the network. Consequently, a convolutional channel's importance is assessed independently of other channels. *The progress of saliency is tracked for each channel* from the beginning of the training process. At each training step, a reinforcement learning algorithm is employed to *select a subset of channels to activate based on the gathered saliency information.* Then, *a forward and a backward pass are run only on these loaded activated channels*, hence executing them and learning their parameters. After identifying the most salient convolutional channels, I *fine-tune their weights* after freezing the network topology.

The contributions of this chapter can be summarized as follows. Firstly, **I propose a dynamic channel execution methodology** which enables training compact models directly by integrating the channel selection procedure in the training process. In principle, the computational graph of the compact model in each training step comprises only active channels, therefore **the memory footprint and computational cost of training are reduced**. The proposed framework also gives direct control over the percentage of channels to activate and execute. Secondly, I **formulate the channel selection problem as a combinatorial multi-armed bandit problem**, and propose to use the combinatorial

upper confidence bound algorithm (CUCB) [23] to solve it. In the proposed algorithm, I make an assumption about model architecture, that is the exact connectivity pattern of a network can be disregarded and we can treat a network as a "bag-of-channels", similarly to the "bag-of-words" assumption in language models. This assumptions makes the optimal channel selection procedure tractable. An advantage of the proposed method is that *additional model complexity is not required to implement the channel selection procedure* unlike [132]. Experiments demonstrate the proposed procedure significantly reduces the cost of training neural networks. For instance, a ResNet-50 network trained on the SVHN dataset with 30% active channels outperforms the baseline with over 4× parameter count reduction and 3× floating-point operations reduction.

4.1.1 Related work

Adaptive computation aims to change the network topology dynamically at run-time. The main focus has been reducing computational cost at inference [232, 242, 238, 128]. More recently, [24] has investigated input-dependent dynamic filter selection. Their framework uses a backbone network, which performs the actual prediction, and a gater network, which decides which filters in the backbone to use. Sparsity in the number of gates, or active filters, is encouraged via L1 regularization. Y. Su and S. Zhou et al. [212] propose a dynamic inference method which provides various inference path options. This is achieved by dividing the original network models in blocks and utilizing a gating mechanism to predict the on/off status of each block during inference. In comparison to the method proposed in this chapter, both of these methodologies *introduce additional model complexity* to achieve gating, and are *not designed for efficient training on resource-constrained applications* but rather only for inference. The closest work to ours is Dynamic Deep Neural Networks [132] which allows selective execution by integrating backpropagation with reinforcement learning. The proposed framework uses Q-learning to learn the parameters of control nodes that influence the computational path. The similarity of this setup to the method proposed in this chapter is that control nodes, and therefore certain network modules, can be on or off during training as well as inference, however, *Dynamic Deep Neural Networks also require extra parameters* for the control nodes, and their method *is input-dependent*.

Pruning is applied after training an initial network in order to eliminate model parameters, therefore making inference less computationally intensive, and decreasing model size and memory footprint. Additionally, pruning can improve generalization if seen through the lens of regularizing an overparameterized model. One approach is non-structured pruning which dates back to Optimal Brain Damage [122]. More recently, Han et al. [86] propose to prune individual weights with small magnitude, and Srinivas and Babu [209] propose to

remove redundant neurons iteratively. The issue with non-structured pruning is that it requires specialized hardware [87] to run. Structured pruning [257, 27, 153, 236, 134, 126], on the other hand, does not require dedicated libraries/hardware as it prunes whole filters, channels of convolutional kernels or even layers, based on some importance criteria. Other post-processing techniques which produce compact network models include knowledge distillation [92, 184], weight quantization [181, 33], low-rank approximation of weights [121, 39]. *All of these approaches are aimed at creating compact networks for inference* as opposed to reducing the compute requirements for training, which is the main goal of work introduced in this chapter.

The Lottery Ticket Hypothesis. Literature has been able to show that a fully trained model can be pruned to under $5 - 10\%$ of its original size with practically no performance degradation. What had been long impossible is successfully training sparse networks from scratch. However, Frankle and Carbin [64] observe that a simple pruning procedure can in fact unravel a subnetwork within an overparameterized model which when trained again in isolation does match the test accuracy of the original network. The key insight is that the weights of the sparse network have to be re-initialized to their original values as they were prior to training the overparameterized model. Identifying a 'winning ticket' was as simple as applying (iterative) magnitude pruning to an already trained dense model. The remaining unpruned connections consitute the architecture of the 'winning ticket'. In their original paper, the authors find winning tickets smaller than $10 - 20\%$ of the size of network architectures for MNIST and CIFAR10. Scaling the procedure to deeper network architectures is trickier because it requires adapting the learning rate schedule [65]. Follow up research has focused on the effect of fine-tuning instead of re-initializing the weights of the sparse networks, or rewinding the weights to a a previous iteration, adapting the learning rate [182], [254], and the effect of data order [66]. A key finding was the effect of 'informed' masking. It views masking the overparameterized model as a form of accelerated training, which simply speeds up the optimization trajectory of weights that were already headed toward 0. This can be seen as an inductive bias, and opens up the option of not training the network weights themselves but rather finding the right mask. Zhou et al. (2019) [254] show that it is possible to learn the mask by making it differentiable and training it with a REINFORCE-style loss. You et al. [247] observe that pruning masks change during initial stages of training but converge quickly (hence coin the term 'early-bird' tickets). Based on this empirical observation, they propose to detect early-bird tickets by calculating the Hamming distance of pruning masks between consecutive pruning iterations. If the distance is small, the pruning mask is fixed, and the ticket can simply be retrained to restore performance. Finally, a noteworthy finding is generalizing the Lottery Ticket Hypothesis to natural

language processing and reinforcement learning. Yu et al. [250] find that it is possible to prune 2/3 of all transformer weights while still achieving strong performance. For reinforcement learning, different pruning levels were needed to achieve strong performance on a task.

4.2 Methodology

In this chapter, I consider a supervised learning problem with a set of training examples $\mathcal{D} = \{\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}, \mathbf{Y} = \{y_1, y_2, \ldots, y_N\}\}$, where $\mathbf{x}$ and y represent an input and a label, respectively. Given a CNN model with L convolutional layers, let each layer $l \in 1 \ldots L$ comprise K_l channels, C_l^k, where $k \in 1 \ldots K_l$ is the channel index. In each training step t, the proposed method:

1) samples a batch of B data samples ($\mathbf{x}^{1:B}$, $\mathbf{y}^{1:B}$);

2) selects and activates a subset S of convolutional channels (see Figure 4.2)

3) runs a forward and a backward pass on the "thin" network, that is only on the active channels;

4) observes the revealed saliency estimates ($\text{SAL}_{l,t}^k$) of the activated channels.

The saliency metric I employ was proposed by Molchanov et al. [153] and approximates the change in loss incurred from removing a particular channel. I maintain an empirical channel saliency mean $\hat{\mu}_l^k$ as well as the number of times T_l^k each channel has been activated in all training steps so far. More precisely, if channel C_l^k has been activated T_l^k times by the end of training step t, then the value of $\hat{\mu}_l^k$ at the end of training step t is $(\sum_1^t SAL_{l,t}^k)/T_l^k$. In addition, it is assumed that the cardinality of S, that is the number of channels to be activated in each training step, is predefined and set beforehand. For instance, I can decide to only activate and execute 20% of all channels at run-time, which would be 1100 channels for the VGG-19 network as an example. After training the network for a given number of iterations, the subset of active channels is fixed by selecting the top percentile according to their global rank of mean saliency estimates across all layers, and fine-tune the model. This final fine-tuning stage can be viewed as operating solely in exploitation mode.

Table 4.1: Notation commonly used in this chapter.

Symbol	**Definition**
$\mathbf{x}$	(image) input
y	label
L	number of convolutional layers in network
$l \in 1 \cdots L$	convolutional layer index
K_l	number of convolutional channels in layer l
$k \in 1 \cdots K_l$	channel index
C_l^k	Channel k in layer l
SAL_l^k	Saliency estimate for channel C_l^k
$\hat{\mu_l^k}$	empirical saliency mean for C_l^k
$\bar{\mu_l^k}$	adjusted channel saliency
T_l^k	times channel C_l^k has been activated
S	subset of activated channels

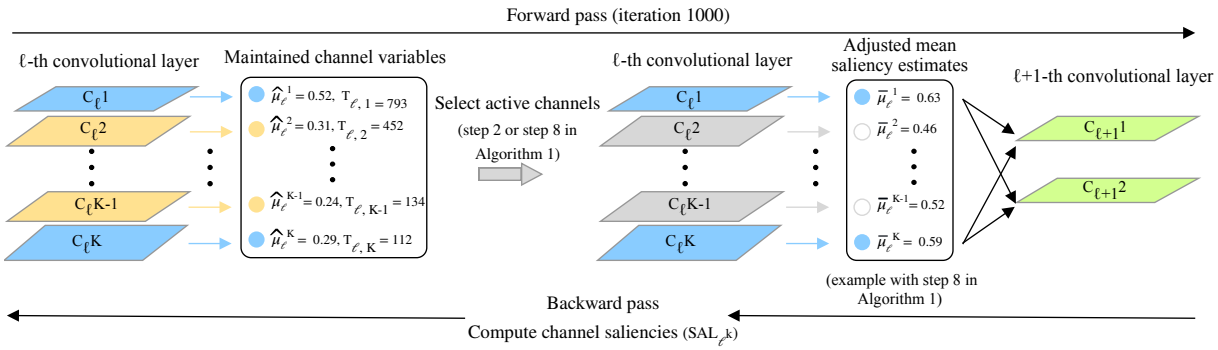


Figure 4.2: I associate two channel variables with each convolutional channel C_l^k: a mean saliency estimate, $\hat{\mu}_l^k$, and a variable which stores the number of times the channel has been activated in all training steps so far, T_l^k. The proposed framework selects a number of channels to activate according to the combinatorial upper confidence bound algorithm. After initializing the channel variables via randomly sampling channels for a number of training steps, I use $\hat{\mu}_l^k$ and T_l^k to calculate the adjusted mean saliency estimates $\overline{\mu}_l^k$. The channels with the highest adjusted saliency estimates will be activated (in blue colour), whereas the rest (yellow) will be not be part of the compact network (grayed out) in the current training step.

4.2.1 Dynamic channel selection mechanism

The proposed methodology for dynamic channel selection is based on the combinatorial upper confidence bound algorithm (CUCB) [23], which is a general framework for the combinatorial multi-armed bandit problem (CMAB). In the CMAB setting we are given a system of arms (or resources/ machines) each having unknown utility, that is an unknown distribution of reward with an unknown mean μ. The task is to select a combination of arms so as to minimize the difference in total expected reward between always playing the optimal super-arm (the combination of arms), and playing super-arms according to the CUCB algorithm. The modified version of the CUCB framework applied to dynamic channel selection is summarized in Algorithm 1. In this version of the algorithm each network channel can be viewed as an arm, whereas a super-arm constitutes the subset of channels S to be activated at each training step.

Algorithm 1 can be loosely divided in two stages: firstly, an initialization round of exploring the saliencies of the channels in the network, and a second stage where the algorithm starts exploiting and refining the initial saliency estimates so as to guide the channel selection procedure. The initialization stage comprises steps 1 and 2, whereas stage 2 comprises steps 4 to 11 of Algorithm 1. In the first stage, the algorithm iterates over all channels in the network, and for each channel C_l^k, activates an arbitrary set of channels S, such that $C_l^k \in S$. Then, the "thin" network is run on a batch of training examples $(\mathbf{x}^{1:B}, y^{1:B})$, and the saliency estimates SAL_l^k of the activated channels are

observed. This first stage is required to initialize the channel saliency means $\hat{\mu}_l^k$ and the number of times each channel has been executed T_l^k, such that these variables can be used to drive the dynamic channel selection mechanism. The second stage of the framework differs in how the active channels are selected. Instead of randomly sampling an arbitrary set of channels, the *adjusted* channel saliencies $\overline{\mu}_l^k$ are calculated, and the channels with the highest adjusted saliencies at each training step are activated. More specifically, the adjusted channel saliency $\overline{\mu}_l^k$ for channel C_l^k which has been activated T_l^k times by the end of training step t is $\overline{\mu}_l^k = \hat{\mu}_l^k + \sqrt{\frac{3 \ln t}{2T_l^k}}$.

The main motivation for choosing the CUCB algorithm for dynamic channel activation is the infeasibility of other approaches used for adaptive computation. For example, I could impose channel sparsity via L1-regularization, but this would require that the entire network is loaded in GPU memory to run a forward pass. Channel gating on the other hand, and to an even greater extent Q-learning, would introduce additional learnable parameters, which goes against the main goal of parameter-efficiency.

In the proposed algorithm, I make the assumption that I can disregard the connectivity pattern and treat the network as a "bag-of-channels", similarly to the "bag-of-words" assumption in language models. First, this independence assumption simplifies the calculation of the channel saliency metric. Second, it allows for a strategy for choosing which channels to activate at each training step. Because of the channel independence assumption, I model the expected reward of activating a selected subset of channels S at training step t, $R_t(S)$, as the summation of the expected rewards of individual convolutional channels. I denote the reward of a given channel as r, so $E[R_t(S)] = E[\sum_{i \in S} r_i] = \sum_{i \in S} E[r_i] = \sum_{i \in S} \mu_i$. Hence, to maximize the expected reward $R_t(S)$ under the channel independence assumption, we should active the channels with the highest saliency as this will maximize the final summation. Since the algorithm does not have access to the true channel reward means μ, I use the adjusted saliency means, $\overline{\mu}$, as an estimate to drive the channel selection procedure. Adjusting the expected channel saliencies in Algorithm 1 can be seen as encouraging exploring infrequently activated channels in earlier training steps. This effect is decayed in later rounds.

In my implementation of the algorithm, I use masking, i.e. multiplying the output of inactive channels by zero, to sparsify the networks. Alternatively, I could instantiate a new compact model only consisting of active channels and copy the corresponding weights from the original sparse network. This would have to be performed at each training step to benefit from a lower GPU memory footprint and floating-point operations at run-time.

Algorithm 1 The CUCB algorithm applied to selective neural network channel execution

For each channel k in each convolutional layer l maintain: 1) T_l^k as the total number of times the particular channel has been activated so far; 2) $\hat{\mu}_l^k$ as the mean of all saliency estimates observed so far.
For each channel k in each convolutional layer l, activate an arbitrary set of channels S, such that $C_l^k \in S$, run forward and backward passes through "thin" network, and update T_l^k and $\hat{\mu}_l^k$.
$t \leftarrow$ number of convolutional channels in network
for j=1...J **do**
  for batch ($\mathbf{x}^{1:B}$, $y^{1:B}$) in $\mathcal{D}$ **do**
    $t = t + 1$
    For each channel C_l^k, set $\overline{\mu}_l^k = \hat{\mu}_l^k + \sqrt{\frac{3 \ln t}{2 T_l^k}}$
    $S \leftarrow$ top percentile of channels according to $\overline{\mu}_l^k$
    Activate channels in S
    Run forward and backward passes through "thin" network
    Update all T_l^k and $\hat{\mu}_l^k$
  end for
end for
$S \leftarrow$ top percentile of channels according to $\hat{\mu}_l^k$
Activate channels in S
Fine-tune final "thin" network

4.2.2 Estimating channel saliency

The proposed dynamic channel selection framework requires the estimation of channel saliency, that is the contribution of each active channel to the overall network performance. The framework needs a saliency metric which enables the algorithm to rank the filters of the entire network globally, that is across layers. Molchanov et al. [153] propose a pruning method which leverages a first-order Taylor approximation for global channel ranking, whereas Theis et al. [226] use Fisher information to achieve kernel ranking across layers [55]. Our channel ranking approach is based on Molchanov et al. [153] although both methods would be applicable.

The intuition behind the approach is approximating the change in the loss function from removing a particular channel, which was active at training step t, via a first-order Taylor expansion. Let h_l^k be the feature map produced by the channel C_l^k. Also, let $\mathbf{b}$ denote a single batch ($\mathbf{x}^{1:B}$, $y^{1:B}$) from the training data $\mathcal{D}$. Then $L(\mathbf{b}, h_l^k)$ will denote the loss of the network when channel C_l^k is active. On the other hand, I will use the notation $L(\mathbf{b}, h_l^k = 0)$ to denote the loss of the network had C_l^k been inactive at the given training step.

$$\left|\Delta L(h_l^k)\right| = \left|L(\mathbf{b}, h_l^k = 0) - L(\mathbf{b}, h_l^k)\right| = \hat{SAL}_{l,t}^k \tag{4.1}$$

Intuitively, the salience metric considers channels which incur a high increase in loss when deactivated as more important. After a first-order approximation around the point $h_l^k = 0$, Eq. 4.1 can be shown to be equal to:

$$\hat{SAL}_{l,t}^k = \left| \frac{1}{M} \sum_{m=1}^{M} \frac{\delta L}{\delta h_{l,m}^k} h_{l,m}^k \right|, \tag{4.2}$$

where M is the length of the vectorized feature map. Eq. 4.2 assumes independence between channel parameters. The computation of Eq. 4.2 is easy in practice as it only requires the first derivative of the loss w.r.t each feature map element, i.e. $\frac{\delta L}{\delta h_{l,m}^k}$, which is computed during backpropagation. In order to compare the saliencies of the activated channels after each training step, I normalize the raw values as their scale varies across layers. I use l_2-normalization before ranking the saliencies:

$$SAL_{l,t}^k = \frac{\hat{SAL}_{l,t}^k}{\sqrt{\sum_j (\hat{SAL}_{l,t}^j)^2}} \tag{4.3}$$

The saliency metric is used to update the mean saliency estimates $\hat{\mu}_l^k$ of all activate channels.

4.3 Experiments

In this section I empirically demonstrate the effectiveness of the framework for dynamic channel selection during training. I conduct all experiments in PyTorch [172].

4.3.1 Datasets

I evaluate the performance of the proposed method on three datasets: CIFAR-10, CIFAR-100 [116] and Street View House Number (SVHN) [161]. Both CIFAR datasets consist of coloured natural images with resolution 32×32, and comprise 50,000 training examples and 10,000 testing examples. CIFAR-10 is drawn from 10 classes, whereas CIFAR-100 from 100 classes. The SVHN dataset consists of 32×32 coloured digit images. I use all 604,388 training images from SVHN and validate on the test set of 26,032 images. I normalize the input data using the channel means and standard deviations. The data augmentation scheme I use consists of padding the images with 4 on each border, then randomly cropping and horizontally flipping a 32×32 patch.

4.3.2 Network models

I evaluate the method for dynamic channel selection on three network architectures: VGGNet [201], ResNet [89] and DenseNet [100]. I use a VGG-19 architecture of type "VGG-E", however I use only a single fully connected layer (16conv + 1FC) [25]. For ResNet I employ a 50-layer architecture with a bottleneck structure (ResNet-50). For DenseNet I use a 121-layer architecture with a growth rate of 32 (DenseNet-121). I use batch normalization on all network models and remove all dropout layers. I do not apply the dynamic channel selection procedure on fully connected layers as they comprise only the final classification layer in the models.

4.3.3 Training and Fine-tuning

All network models are trained using SGD with Nesterov momentum of 0.9 without dampening. The minibatch size I use is 64 for all datasets. On the two CIFAR datasets, I train for 160 epochs, and on the SVHN dataset for 20 epochs. The initial learning rate is set at 0.1 and is divided by 10 at 50% and 75% of the training epochs. I also use a weight decay of 10^{-4}. I adopt the weight initialization introduced by [88]. After training following the proposed framework, I fix the subset of active channels by selecting the most salient ones across all layers. I then build a compact model consisting only of active channels and copy the corresponding weights. This compact model is fine-tuned by repeating the training procedure. I evaluate the floating-point operations (FLOPs) and the number of parameters of the compact networks using a PyTorch operations counter package [256].

4.3.4 Results

I evaluate the dynamic channel selection framework on all combinations of datasets and network architectures. I experiment with a ratio of active channels between 20% to 100% in 10% increments. I perform all experiments three times and report averaged results. Tables 4.2, 4.3 and 4.4 show the highest test accuracies achieved by the models in **boldface**.

Table 4.2: Test Accuracy on CIFAR-10

Model	Test accuracy	Parameters	% Baseline	FLOPs	% Baseline
VGG-19(Baseline)	92.86%	20.035×10^6	-	3.98×10^8	-
VGG-19(70% channels)	**93.11**%	9.57×10^6	48%	2.84×10^8	71%
VGG-19(40% channels)	92.02%	2.89×10^6	14%	1.32×10^8	33%
ResNet-50(Baseline)	92.94%	23.52×10^6	-	0.85×10^8	-
ResNet-50(90% channels)	**93.52**%	20.96×10^6	89%	0.74×10^8	87%
ResNet-50(30% channels)	92.93%	5.48×10^6	23%	0.3×10^8	35%
DenseNet-121(Baseline)	93.22%	6.96×10^6	-	0.59×10^8	-
DenseNet-121(80% channels)	**93.73**%	5.14×10^6	74%	0.44×10^8	75%
DenseNet-121(40% channels)	92.82%	1.37×10^6	20%	0.13×10^8	22%

Table 4.3: Test Accuracy on CIFAR-100

Model	Test accuracy	Parameters	% Baseline	FLOPs	% Baseline
VGG-19(Baseline)	68.37%	20.08×10^6	-	4×10^8	-
VGG-19(90% channels)	**70.31**%	16.16×10^6	80%	3.77×10^8	94%
VGG-19(40% channels)	66.72%	2.85×10^6	14%	1.48×10^8	37%
ResNet-50(Baseline)	72.88%	23.7×10^6	-	0.86×10^8	-
ResNet-50(90% channels)	**73.04**%	21.6×10^6	91%	0.75×10^8	87%
ResNet-50(40% channels)	70.34%	8.45×10^6	36%	0.4×10^8	47%
DenseNet-121(Baseline)	74.08%	7.05×10^6	-	0.6×10^8	-
DenseNet-121(90% channels)	**75.07**%	6.21×10^6	88%	0.52×10^8	87%
DenseNet-121(40% channels)	70.98%	1.4×10^6	20%	0.14×10^8	23%

Regularization effect. For all datasets and network models, the highest test accuracy is achieved when the percentage of active channels is typically between 70%-90%. The only

Table 4.4: Test Accuracy on SVHN

Model	Test accuracy	Parameters	% Baseline	FLOPs	% Baseline
VGG-19(Baseline)	95.57%	20.035×10^6	-	3.98×10^8	-
VGG-19(80% channels)	**96.04**%	12.64×10^6	63%	3.22×10^8	81%
VGG-19(40% channels)	95.52%	2.81×10^6	14%	1.43×10^8	36%
ResNet-50(Baseline)	92.34%	23.52×10^6	-	0.85×10^8	-
ResNet-50(30% channels)	**94.39**%	5.49×10^6	23%	0.27×10^8	32%
DenseNet-121(Baseline)	94.07%	6.96×10^6	-	0.59×10^8	-
DenseNet-121(90% channels)	**94.8**%	6.11×10^6	88%	0.51×10^8	86%
DenseNet-121(40% channels)	94.31%	1.36×10^6	20%	14.4×10^8	24%

exception is ResNet-50 evaluated on the SVHN dataset which achieves peak classification when 30% of the channels are active. I hypothesize that the increase in accuracy is due to the regularization effect of the dynamic channel selection procedure which could be viewed as feature selection applied on the hidden layers. A similar phenomenon is observed in the pruning frameworks [134], where L1 regularization is imposed on channels instead, as well as [236, 240]. Finally, it is noteworthy to mention that the proposed framework outperforms Dynamic Deep Neural Networks [132] in terms of relative drop in accuracy compared to baseline for ResNet evaluated on CIFAR-10. Dynamic Deep Neural Networks observe up to 7% drop in accuracy whereas our framework maintains baseline performance even with 20% active channels (see Figure 4.3).

Parameter and FLOP reduction. The main goal of the dynamic channel execution framework is to lower the demands for training parameter-efficient networks. Since the best performing models use active channels between 70%-90%, their parameter count and FLOPs are also lower compared to baseline models. For example, on CIFAR-10, the best performing VGG model achieves 2× parameter reduction. DenseNet-121 and especially ResNet-50 are unable to achieve such parameter-efficiency owing to their bottleneck architecture. More specifically, the skip connections in ResNet-50 based on the addition operation require that certain convolutional layers have the same number of output channels. In tables 4.2, 4.3, 4.4 I have also shown results from smaller networks with 30%-40% active channels (last row of each model). These very compact networks demonstrate comparable accuracy to baseline on the CIFAR-10 and SVHN datasets with parameter reduction between 3×-7× and FLOPs reduction 2×-5×. Moreover, ResNet-50 can achieve over 9× parameter reduction on CIFAR-10 and SVHN (see Figures 4.3, 4.5) while maintaining baseline-level performance. Nevertheless, the small networks perform worse

than baseline on the CIFAR-100 dataset (2%-3% accuracy drop for models with 30%-40% active channels). I conjecture this is because CIFAR-100 contains 100 classes and extra model capacity is required.

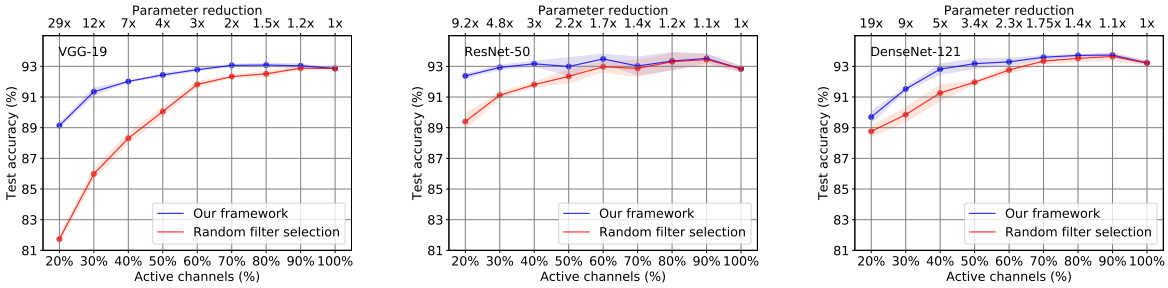


Figure 4.3: CUCB vs random channel selection (CIFAR-10)

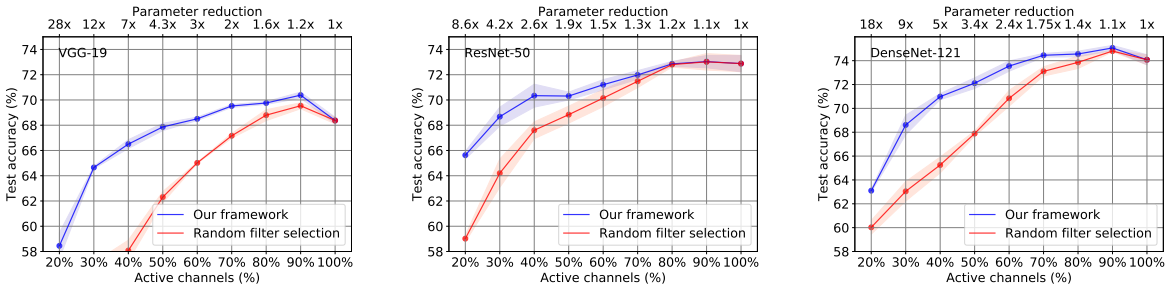


Figure 4.4: CUCB vs random channel selection (CIFAR-100)

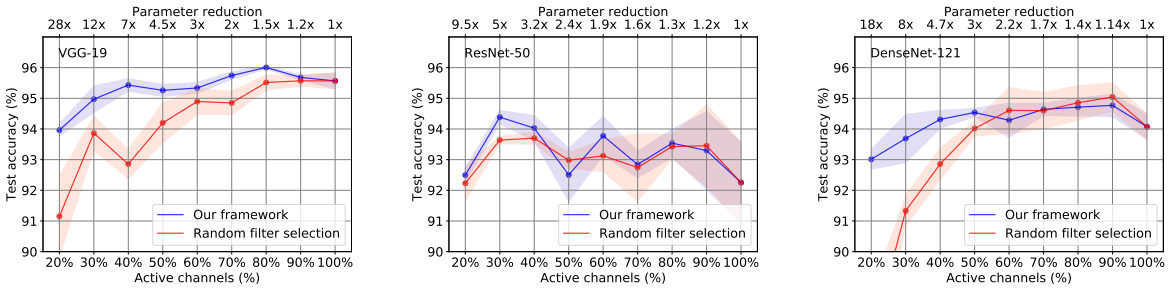


Figure 4.5: CUCB vs random channel selection (SVHN)

Results on CUCB vs random channel selection. I performed experiments where instead of activating the channels with the highest mean estimated saliency prior to fine-tuning (step 12 in Algorithm 1), I randomly select a subset of channels to activate (shown in red in Figures 4.3, 4.4, 4.5). It can be observed that randomly selecting channels to fine-tune performs worse in general than the proposed methodology. The difference in performance becomes more significant as the active channels become fewer. The sequential VGG-19 architecture adheres to this behaviour on all datasets. For ResNet-50 and DenseNet-121 the performance of models fine-tuned on randomly selected channels is on occasion on par with the proposed framework. More specifically, on CIFAR-10

when the active channels are $\geq$ 70%, and on SVHN when random channel selection even outperforms our framework on a few instances. I hypothesize the methodology, which is based on the assumption of independence between channels, might be adversely affected by architectures with skip connections across layers, such as ResNet and DenseNet.

Single-stage training. Instead of following the proposed two-stage process of firstly training a model with the dynamic channel selection framework, and then fine-tuning by repeating the same training procedure on the most salient channels, I attempt an integrated pipeline. I experiment with the VGG-19 network on CIFAR-10 with the same hyperparameter settings as described in Section 3.3. For the first 40 iterations I use the proposed dynamic channel selection framework, then I activate the most salient channels and freeze the network topology for the remainder of the training process. I compare the performance of the single-stage and two-stage training procedures in Table 4.5. Overall, a small degradation in performance for the integrated training procedure can be observed with a test accuracy drop between 0.18%-0.86%.

Active channels (%)	20%	30%	40%	50%	60%	70%	80%	90%
Single-stage test accuracy (%)	89.4	90.98	91.15	91.74	91.96	92.2	91.91	92.24
Accuracy gain (%)	+0.26	-0.36	-0.86	-0.7	-0.82	-0.86	-0.18	-0.8

Table 4.5: The test accuracy performance for VGG-19 on CIFAR-10 with a single-stage training process. Accuracy gain (- sign means accuracy drop) is compared to the equivalent two-stage training procedure.

4.4 Discussion

In this section I provide additional discussion on the parameter independence assumption of the proposed method and how it compares to other pruning approaches, as well as provide additional considerations about the system implementation of the method.

Parameter independence assumption. I place a strong assumption on parameter independence. Although the exact connectivity pattern clearly impacts performance, this simplifying assumption brings advantages by making the calculation of channel saliency tractable and results in a simple strategy to maximizing the expected reward by activating the top salient channels. In fact, connectivity pattern is not taken into account in other pruning approaches either, including magnitude-based pruning, which is used in works

related to the Lottery Ticket Hypothesis. We can demonstrate this by looking at the Taylor expansion around the point when a parameter w_i is 0:

$$L(\mathbf{w} + \delta\mathbf{w}) - L(\mathbf{w}) \approx \frac{1}{2}\delta\mathbf{w_i}^T H \delta\mathbf{w_i} \tag{4.4}$$

The model perturbation is chosen such that we only zero out a single weight w_i and we assume the model is trained to a local minimum so the gradient term can be ignored. H is the Hessian matrix. Since we are interested in weights that will not incur any change of loss when perturbed, we can derive the following saliency measure by minimizing Eq. 4.4:

$$SAL_i = \frac{\mathbf{w_i}^2}{2[\mathbf{H}^{-1}]_{ii}} \tag{4.5}$$

where $[\mathbf{H}^{-1}]_{ii}$ denotes the i^{th} diagonal element of the inverse Hessian matrix of the loss L of the given model. After sorting the metric in decreasing order, we can prune the network by removing the lowest value weights as measured by this saliency metric. Computing the inverse of the full Hessian is not practical for bigger network models. Let us assume for practicality that the Hessian is the identity matrix, possibly rescaled by a constant. This would mean that the Hessian matrix is diagonally-dominant, and that its diagonal entries are roughly uniformly distributed. In this case, the proposed saliency metric would be equivalent pruning the weight of lowest magnitude. Recall that in Eq. 4.1, 4.2 and 4.3, I follow an identical justification for deriving saliency but applied to the feature maps. I implicitly make the same assumptions about the structure of the Hessian matrix in my derivation, which is also the root cause of the parameter independence assumption. We can conclude my proposed method does not make any stronger assumptions than other works relying on magnitude-based pruning. An in-depth review of sparsity in deep learning is provided in [93].

Relation to the Lottery Ticket Hypothesis. A pertinent question to ask is if the methodology proposed in this chapter can be reformulated as a strategy to find winning tickets. The closest version of the Lottery Ticket Hypothesis to my proposed method is by You et al. [247] who make the interesting observation that pruning masks applied to entire convolutional channels change drastically only in the first epochs of training. The authors propose to find winning tickets by pruning $p\%$ of the weights after each training iteration and calculating a mask distance at each step. When this distance falls below a threshold, the mask can be fixed, the weights re-initialised and the winning ticket can be trained from scratch. The main differences between the method in this chapter and [247] are: (1) my method never runs the full dense model; (2) I fine-tune the weights instead of re-initializing them. It is possible that a winning ticket could be identified after several burn-in epochs of the CUCB algorithm, then the top percentile of channels are activated,

re-initialized to their original values and trained from scratch. This experiment is left outsite of this chapter as future work.

Systems performance. The implementation of the algorithm uses binary masking applied to the convolutional filters to achieve structured sparsity. On the other hand, the parameter and FLOPs estimates provided in the results section assume that the weights have in fact been deleted from the computational graph of the network when running the forward and backward passes. The proposed algorithm requires that the computational graph be reloaded at each forward pass as well as weights moved between the device running the network and some memory external to it. These operations would add significant computational overhead to the algorithm. A viable implementation would depend on having high bandwidth data exchange between the low-memory GPU and the external memory containing all parameter values and their saliencies, as well as an efficient procedure to instantiate a new computational graph representing the new connectivity pattern at each training step.

4.5 Summary

In this chapter I have proposed a novel methodology, inspired by the combinatorial multi-armed bandit problem, to dynamically identify and utilize only the most salient convolutional channels at each training step. As a result, the memory and computational burden during both training *and* inference is reduced. The proposed method tracks the relative importance of each channel, and at each training step, a subset of highly salient channels are activated and executed according to the combinatorial upper confidence bound algorithm. The algorithm is supported by an assumption about parameter independence of the network model. Experimental results on several datasets and state-of-the-art network architectures reveal the framework is able to significantly reduce computational cost (up to $4\times$) and parameter count (up to $9\times$) with little or no loss in accuracy. I also demonstrate the proposed framework consistently performs better than random channel selection.

Chapter 5

Inductive biases in probabilistic modelling for dynamic graphs

Representation learning over graph-structured data has generated significant interest owing to widespread application in a variety of interaction-based networks, such as social and communication networks, bio-informatics and relational knowledge graphs. Developing methods for unsupervised graph representation learning is challenging as it requires summarizing the graph structural information in low-dimensional embeddings. These representations can then be used for downstream tasks, such as node classification, link prediction and community detection.

The majority of unsupervised graph representation learning methods have focused solely on static non-evolving graphs, while many real-world networks exhibit complex temporal behaviour. The methods aimed at dynamic graphs focus solely on node evolution [75, 76, 193, 255]. However, they do not lend themselves to capturing the evolution of graph-level structures, such as clusters of nodes, or communities. On the other hand, the patterns of evolving node clusters are of great interest in social networks [115, 77, 243], as well as encountered in the temporal organization of large-scale brain networks [234], among others.

To address this challenge, in this chapter I propose GRADE (GRAph Dynamic Embedding) - a probabilistic generative model for jointly learning evolving node and community representations. The benefit of modelling the interaction between nodes and communities in the static setting was studied by vGraph [215]. Further, [9] produces evidence that taking into consideration higher-order graph structures, such as communities, enhances our capability to model emergent dynamical behaviour. Consequently, in this chapter, I extend the idea of modelling node-community interactions to the dynamic case. I apply principles of efficiency by *encoding prior knowledge about the mechanism of edge formation* in the model, that is nodes are more likely to connect to nodes in the same community,

as well as *assuming smoothness in the temporal evolution of the node and community embeddings*. I also derive a parameter-efficient inference procedure to learn from data.

In summary, the key principles of efficiency employed in this work are as follows:

(1) I introduce node and community dynamics by assuming a smooth temporal trajectory and evolving their respective representations as Gaussian random walks.

(2) The assumed edge generative mechanism incorporates an inductive bias about the existence of community structure, which is common in real-world networks.

(3) I propose an efficient variational inference procedure to learn the parameters of the model. The use of neural networks to implement the generative model facilitates the use of amortized inference, which keeps the number of model parameters low.

In this chapter, first I compare the proposed method, GRADE, against other works on dynamic graph representation learning. Then, I proceed to describe the assumed dynamic graph generative process as well as the inference procedure. Finally, I experimentally validate the efficacy of this method on real-world datasets.

5.1 Literature review

5.2 Overview of traditional clustering methods

There is vast literature on graph clustering. A comprehensive overview is provided in [62, 60] First, I present a summary of traditional methods on the topic: graph partitioning, hierarchical clustering, partitional and spectral clustering, methods based on statistical inference, and optimization-based methods. The problem of *graph partitioning* is to divide the nodes in clusters of a predefined size. The goal of partitioning is to minimize the number of edges between clusters and most variants of this approach are NP-hard although some techniques have shown good results despite note being optimal. A good example is the Kernighan–Lin algorithm [107, 213], which performs iterative bisections to cluster. Another example is spectral bisectioning, which is based on the properties of the spectrum of the Laplacian matrix and the min-cut theorem [59], which can be used to determine minimal cuts from maximal flows and identify communities in graphs [72, 57, 58]. The problem with graph partitioning is that we need to preassign cluster sizes, and we are unlikely to have access to such knowledge. On the other hand, if the graph has a hierarchical structure, where each cluster is composed of smaller ones, this would be a perfect use case for *hierarchical clustering*. The first step to applying hierarchical clustering is coming up with a measure of node similarity. Such similarity metrics include structural

equivalence [18, 239, 135], independent paths between vertices [48, 50, 51], properties of random walks on graphs, e.g. commute time [191, 63, 246, 245]. After computing a similarity matrix, a distance metric can be used to assign highly similar nodes to the same community. This assignment can happen through the application *agglomerative clustering*, where the clusters are iteratively merged based on their similarity, or *divisive algorithms* where the idea is detect the edges that connect vertices of different communities and remove them. A class of clustering techniques, which does not assume a hierarchical structure nor the size of clusters, but the nodes have to embedded in a metric space, is *partitional clustering*. Then, the nodes are clustered by minimizing a cost function based on node distances. The most popular partitional technique is k-means clustering [138]. *Spectral clustering* methods rely on eigenvectors of (similarity) matrices. The goal is to use the value of the eigenvectors as a mapping to a lower dimensional space for each vertex, and apply a standard technique like k-means clustering in this space [199]. The change of representation induced by the eigenvectors makes cluster structure much more evident. The problem with spectral clustering, however, is that it is computationally expensive unless the graph is sparse although the performance is in fact poor in the sparse network setting [164]. *Statistical inference* methods also provide powerful tools for community detection. A standard approach is fitting a generative model to the graph (an approach also popular in deep learning methods). The stochastic block model (SBM) is the most established generative model of graphs with communities. The SBM assumes a network is characterised by block structure, which means the nodes are partitioned into subgroups (i.e. blocks). The probability of edges depends on the blocks the source and target nodes belong to [54, 96, 202]. Despite its popularity, SBM struggles in today's more complex datasets [130]. Finally, I bring attention to optimisation-based methods, which maximize a function that indicates the quality of the clustering. The most popular metric is modularity [162]. There are issues with modularity, however. Firstly, modularity maximization is NP-hard [16]. Moreover, the measure itself is not fully robust to some edge cases, for example there are high-modularity partitions even in random graphs without clusters [81], and it exhibits a preferential scale for communities [61].

5.3 Extending traditional methods to dynamic graphs

Clustering algorithms used for static graphs are often used for dynamic networks as well. A usual representation of a dynamic graph, which is also used in this chapter, is as a series of time snapshots of static networks. One strategy for this problem scenario is to first detect the community structure for each individual snapshot. Then, we can pair communities of consecutive windows by using some similarity metric, for instance the

Jaccard coefficient [97, 4, 169]. This strategy, however, often struggles on real networks that are noisy, and produces partitions with significant temporal variations. A unified framework where communities are inferred from the combined knowledge of several time steps can circumvent this problem. One such strategy is evolutionary clustering, which finds a tradeoff between cluster partitioning at time t and time $t-1$ [21]. Many static clustering techniques can be reformulated in the evolutionary framework [26, 129]. Consensus clustering is another way to achieve stable dynamic communities by combining multiple snapshots. The goal is reaching a consensus partition that is better fitting than the clustering partitions derived from the static snapshots [119]. Consensus clustering is a difficult combinatorial optimisation problem. There are also other numerous examples of extending static community detection methods to dynamic networks. Graph factorization approaches such as DANE [127] rely on spectral embedding, similarly to static methods like [2, 19, 11]. DANE assumes smooth temporal evolution and models it using matrix perturbation theory. Skip-gram models where node representations are learnt by random walk objectives [78, 175, 221] can also be extended with temporal constraints based on time-stamped edges, for example CTDNE [165] and NetWalk [251].

5.4 Deep unsupervised learning on dynamic graphs

The increasing scale of networks and dimensionality of data demand more powerful techniques for effect and efficient performance [187]. Deep learning has emerged as the most promising solution. Graph convolutional neural networks (GNNs) have become a widely used tool for graph representation learning [112, 233, 17]. GNNs leverage the graph's topological information to learn the pattern of node connectivity, neighbourhoods and subgraphs [Wu et al., 2020]. In addition, GNNs are most resilient to sparsity, which is commonly associated with large-scale graphs. A popular approach to include temporal dependency in GNNs is to introduce a recurrent mechanism. For example, [196] proposes two ways to achieve this goal. One way is to obtain node embeddings via a GNN, which are then fed to an LSTM to learn dynamism. The second approach modifies the LSTM layers to incorporate graph-structured data. A different approach altogether is to evolve the graph convolutional parameters with a recurrent neural network (RNN), such as in EvolveGCN [171], as opposed to the node embeddings, hence addressing issues stemming from rapidly changing node sets between time steps. Alternatively, STGCN [249] avoids using RNNs completely by introducing an efficient ST-Conv layer for faster training with few model parameters. To achieve greater capacity, the authors of Temporal Graph Networks associate a memory cell with each note to store compressed historical information of its previous interactions [186]. To update the memory, the authors use message-passing [46].

The most related body of work to this paper is a category of transductive unsupervised methods applied on temporally discrete and non-attributed dynamic graphs. One such approach is DynGEM [75] which employs deep autoencoders to learn node embeddings but resorts to no recurrent structures for temporal dependency. Instead, time dynamics are injected by re-initializing the parameters of the autoencoder at each time step t with the parameters learnt at $t-1$. Unlike our proposed method GRADE, DynGEM can only handle growing dynamic networks. Another method is dyngraph2vec [76] which is trained to predict future links based on current node embeddings using an LSTM mechanism. DynamicTriad [255] models the process of temporal link formation between vertices with common neighbours, that is triadic closure, and enforces latent space similarity between future connected nodes. Finally, DySAT [193] draws inspiration from the success of attention mechanisms and applies them structurally over local neighbourhoods and temporally over historical representations. It is important to note that DySAT has been demonstrated to consistently outperform other dynamic graph representation learning methods, more specifically DynGEM, Dynamic Triad and dyngraph2vec, and hence serves as our strongest dynamic baseline.

The main advantage of GRADE over these competitive methods is most evident on tasks related to community detection. The inductive bias injected in GRADE that nodes cluster together enables for an efficient community assignment mechanism, which I demonstrate in the experimental section, and outperforms all baselines. I also demonstrate that knowledge about community assignment can improve performance on dynamic link prediction. Also, GRADE can be used to infer the node and community embeddings at future time steps. In comparison, these dynamic methods use representations learnt at the last training step for dynamic prediction.

It is also worth noting that GRADE is related to dynamic topic models [41, 13] in that they represent hidden variable state-space models. Topic models are generative models which assume documents have distributions over the topics and each topic has a distribution over the words. In comparison, in GRADE I assume that each node can be assigned to one of several communities, that is my method assumes a distribution over the communities for each node, and for each community I introduce a distribution over the nodes. Some works like [6, 188] have focused on the shift of word meaning over time while others, such as [41], model the shift in topic proportions in documents. In GRADE, I assume both nodes and communities undergo temporal semantic shift.

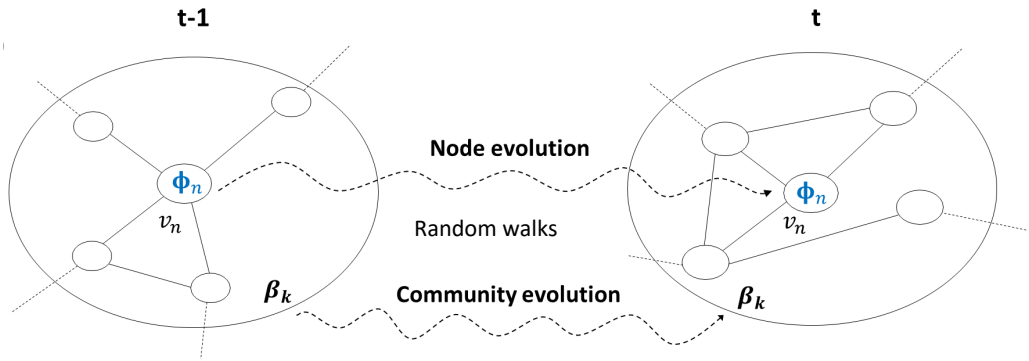


Figure 5.1: In the proposed dynamic graph model, I assume that the semantic meaning of communities *and* the social context of individual nodes evolve over time. This necessitates imposing temporal evolution on both the node and community embeddings, denoted by $\boldsymbol{\phi}$ and $\boldsymbol{\beta}$ respectively, between time steps, which I achieve via Gaussian random walks.

5.5 Methodology

5.5.1 Overview

I represent a dynamic network as a sequence of graph snapshots over a series of discrete and equally-spaced time intervals. At each time step, I model the edge generation process between node neighbours via multinomial community and node distributions. Similarly to vGraph [215], first a community assignment z is sampled for each node v from a distribution over the communities, i.e. $z \sim p(z|v)$. Then, a neighbour c is sampled from the distribution over the nodes of the assigned community, that is $c \sim p(c|z)$. I use learnable transformations of the node and community embeddings, that is neural networks, to produce the parameters of the node and community distributions. In this work, I assume that the semantic meaning of communities *and* the proportions over the communities for each node evolve simultaneously over time. Following an approach introduced in dynamic topic modelling [41], I encode smooth temporal evolution in the method by assuming a Gaussian random walk prior over the representations between time steps. Furthermore, I draw inspiration from social networks where a user's preferences can shift from one community to another. I explicitly model the dynamism in community membership by introducing a node-specific and time-varying transition matrix to update the community mixture coefficients over time. To learn the parameters of the model, I propose an efficient variational inference procedure as calculating the exact posterior is not analytically possible. The use of neural networks to parameterize the node and community distributions, *facilitates the use of amortized inference*. This means I only need to augment the input to the neural networks in order to generate the parameters of the posterior node and community distributions. Further, owing to the choice of the posterior family, I can easily retain the dependence

of the node and community embeddings on their historical states (structured variational inference), which *improves the fidelity of our approximation to the true posterior*. I show that the proposed method *outperforms* both static and dynamic baselines on the tasks dynamic link prediction and dynamic community detection on real-world dynamic graphs extracted from the DBLP bibliography, Wikipedia and Reddit. More specifically, as a probabilistic model, GRADE allows for *explicitly inferring community assignments*, whereas competitive models for dynamic graph representation learning do not possess this capacity. As a result GRADE shows very strong results on the task of dynamic community detection.

The proposed method is aimed at non-attributed dynamic graphs. It is worth noting that although GRADE is a transductive approach, changes of vertex sets between snapshots at different time steps do not pose a problem, if the complete vertex set is known a priori.

5.5.2 Problem Definition and Preliminaries

I consider a dataset comprising a sequence of non-attributed (i.e. without node features) graph snapshots $\Gamma = G^1, \ldots, G^T$ over a series of discrete and equally-spaced time intervals $t \in \{1, \ldots, T\}$, such that t is an integer time index. I assume all the edges in snapshot G^t occur at time t and the complete set of vertices $\mathcal{V} = \{v_1, \cdots, v_N\}$ in the dynamic graph Γ is known a priori. GRADE supports the addition or removal of nodes as well as edges between time steps. I also assume there are K communities (clusters of nodes) in the dynamic network. The proposed method aims to learn time-evolving vector representations $\boldsymbol{\phi}_n^t \in \boldsymbol{R}^L$ for all nodes $v_n \in \mathcal{V}$ and communities $\boldsymbol{\beta}_k^t \in \boldsymbol{R}^L$, $k \in \{1, \ldots, K\}$, for each time step t. Further, a useful model for dynamic graph embedding should not only capture the patterns of temporal evolution in the node and community representations but also be able to predict their future trajectories. Common notation used in this chapter is summarized in Table 5.1.

Table 5.1: Notation commonly used in this chapter.

Symbol	**Definition**
$\Gamma = G^1, \ldots, G^T$	sequence of dynamic graph snapshots
$v_n \in \mathcal{V} = \{v_1, \ldots, v_N\}$	complete vertex set of Γ
$k = 1, \ldots, K$	community indices
$\boldsymbol{\beta}, \boldsymbol{\phi}$	community and node embeddings
γ, σ	temporal smoothness hyperparameters
z	community assignment latent variable
$\boldsymbol{A}_n^t$	transition matrix for node v_n at time t
edge (v, c)	(source node, target node)
$n = 1, \ldots, \lvert V^t \rvert$	vertex set indices
$m = 1, \ldots, deg(v_n)$	neighbour index of v_n
$\boldsymbol{\pi}, \boldsymbol{\theta}$	parameters of community and node distributions

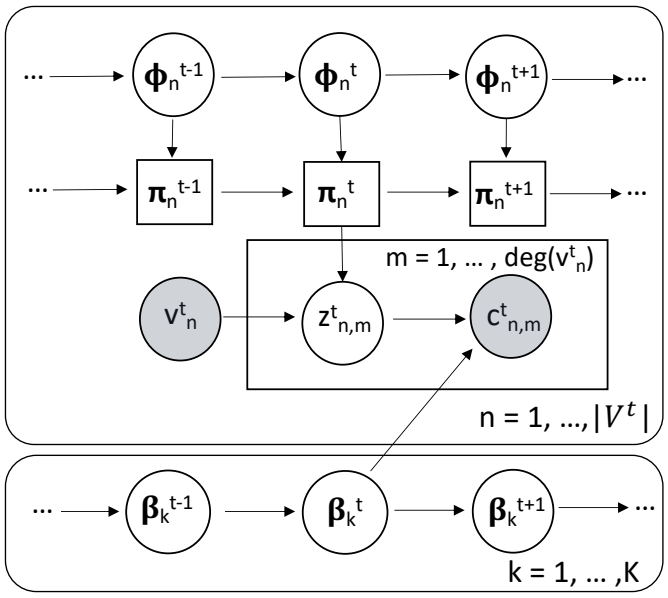


Figure 5.2: Plate notation for the edge generative process of GRADE. The node and community representations, and consequently the parametrization of the node and community distributions, evolve over time. The parameters of the community distribution, $\boldsymbol{\pi}$, are explicitly updated by a deterministic transformation of the node embeddings (denoted by rectangles). The observed data is the edges (v_n, $c_{n,m}$) in the dynamic graph. All notational symbols are described in Table 5.1.

5.5.3 GRADE: Generative Model Description

GRADE is a probabilistic method for modelling the edge generation process in dynamic graphs. I adopt the approach by vGraph [215] to represent each node v in the active vertex set of G^t as a mixture of communities, and each community as a multinomial distribution over the nodes. The linked neighbour generation at each time step t is as follows: first, community assignment z is sampled from a conditional prior distribution over the communities $z \sim p(z|v)$. Then, a neighbour is drawn from the node generative distribution $c \sim p(c|z)$ based on the social context defined by the assigned community. The generative process for graph snapshot G^t can be formulated as:

$$p(c|v) = \sum_{z} p_{\boldsymbol{\theta}^t}(c|z) p_{\boldsymbol{\pi}^t}(z|v) \tag{5.1}$$

where $\boldsymbol{\theta}^t$ and $\boldsymbol{\pi}^t$ parametrize the multinomial generative and prior distributions at time step t respectively. In my dynamic graph model, I suppose that the semantic meaning of communities *as well as* community proportions for nodes change over time. This necessitates capturing the temporal evolution of the underlying node and community distributions by an evolving set of parameters. GRADE achieves this by making these

parameters implicitly dependent on the evolving node and community embeddings, $\boldsymbol{\phi}$ and $\boldsymbol{\beta}$ respectively.

More specifically, I treat the community and node representations as random variables and impose a simple state-space model that evolves smoothly with Gaussian noise between time steps as follows:

$$p(\boldsymbol{\beta}_k^t|\boldsymbol{\beta}_k^{t-1}) = \mathcal{N}(\boldsymbol{\beta}_k^{t-1}, \gamma^2\boldsymbol{I}) \tag{5.2}$$

$$p(\boldsymbol{\phi}_n^t|\boldsymbol{\phi}_n^{t-1}) = \mathcal{N}(\boldsymbol{\phi}_n^{t-1}, \sigma^2\boldsymbol{I}) \tag{5.3}$$

Note that I evolve the embeddings of the complete vertex set $\mathcal{V}$ at each time step, although the model allows for a subset of the nodes to be present at each time step. The temporal smoothness hyperparameters γ and σ control the rate of temporal dynamics. The parametrization of the generative distribution is achieved by first transforming the community representations through a neural network, and then mapping the output through a softmax layer: $\boldsymbol{\theta}_k^t$ =softmax($\zeta(\boldsymbol{\beta}_k^t)$).

To evolve the community mixture weights for nodes, I observe that users' interests in a social network change over time. As a result, users may shift from one community to another. This is characterized by user-specific behaviour within the broader context of community evolution. For these reasons, I explicitly model community transition with a transition matrix. More specifically, for each node v_n, I update the community mixture weights $\boldsymbol{\pi}_n^t$ by means of a $K \times K$ node-specific and time-varying transition matrix $\boldsymbol{A}_n^t$, produced as a function, $\boldsymbol{\psi}$, of the node embeddings:

$$\boldsymbol{\pi}_n^t = \boldsymbol{\pi}_n^{t-1} \cdot \boldsymbol{A}_n^t \tag{5.4}$$

In summary, GRADE's edge generative process for each graph snapshots G^t in $G^1, \ldots, G^T$ is as follows:

1. Draw community embeddings $\boldsymbol{\beta}_k^t$ for $k = 1, \cdots, K$: $\boldsymbol{\beta}_k^t \sim \mathcal{N}(\boldsymbol{\beta}_k^{t-1}, \gamma^2\boldsymbol{I})$
2. Draw node embeddings $\boldsymbol{\phi}_n^t$ for all nodes $v_n \in \mathcal{V}$: $\boldsymbol{\phi}_n^t \sim \mathcal{N}(\boldsymbol{\phi}_n^{t-1}, \sigma^2\boldsymbol{I})$
3. Transition matrix $\boldsymbol{A}_n^t$ is a non-linear transformation, ψ, of the node embeddings: $\boldsymbol{A}_n^t$ = row-softmax $(\psi(\boldsymbol{\phi}_n^t))$
4. Update community mixture coefficients for node v_n: $\boldsymbol{\pi}_n^t = \boldsymbol{\pi}_n^{t-1} \cdot \boldsymbol{A}_n^t$
5. For each edge $(v_n^t, c_{n,m}^t)$ in G^t:

(a) Draw community assignment z from multinomial prior over the communities:
$z^t_{n,m} \sim p_{\boldsymbol{\pi}^t_n}(z|v^t_n)$

(b) Parameters of distribution over the nodes is a function of $\boldsymbol{\beta}^t_z$: $\boldsymbol{\theta}^t_z = \text{softmax}(\zeta(\boldsymbol{\beta}^t_z))$

(c) Draw linked neighbour c from node generative distribution for sampled community z:
$c^t_{n,m} \sim p_{\boldsymbol{\theta}^t_z}(c|z^t_{n,m})$

The plate notation and a flowchart of the proposed generative process are shown in Figure 5.1 and 5.3.

One of the key ideas in GRADE is the evolution of the node and community embeddings under a Gaussian state-space model, as shown in Eq. 5.2 and Eq. 5.3. Here, I provide motivation behind this modelling choice. As discussed in Section 3.2 in [69], the evolution of *continuous* variables in state-space models, such as the node and community representations in GRADE, can be decomposed into a deterministic and a stochastic component. Using ϕ as an example:

$$\boldsymbol{\phi}^t_n = f^t(\boldsymbol{\phi}^{t-1}_n) + \boldsymbol{\epsilon} \tag{5.5}$$

where t is the discrete time index. In the prior, for the deterministic component f^t, *I choose the identity matrix* whereas the noise component $\boldsymbol{\epsilon}$ is sampled from $\mathcal{N}(\mathbf{0}, \sigma^2\boldsymbol{I})$, where σ is a temporal smoothness hyperparameter. Admittedly, I could make other design choices, for example making f^t a transformation learnt from the data. *Using the identity matrix showed strong results* so I did not explore other approaches, which would also introduce more complexity. Also, by tuning the smoothness hyperparameter σ, *I give the user direct control over* the assumed rate of *temporal dynamics*. Since the parameters of the distributions over the nodes and communities are time-invariant functions of the embeddings, evolution in the embeddings implicitly controls the temporal smoothness of the node and community distributions.

Updating the community mixture weights for a given node v_n with a transition matrix A^t_n allows to explicitly inspect community transition patterns or enforce a specific structure on the matrix. For example, we could impose a diagonal dominance prior if we believe community transitions happen infrequently. Although this investigation is left as future work, it justifies the choice of introducing a transition matrix to update the mixture coefficients.

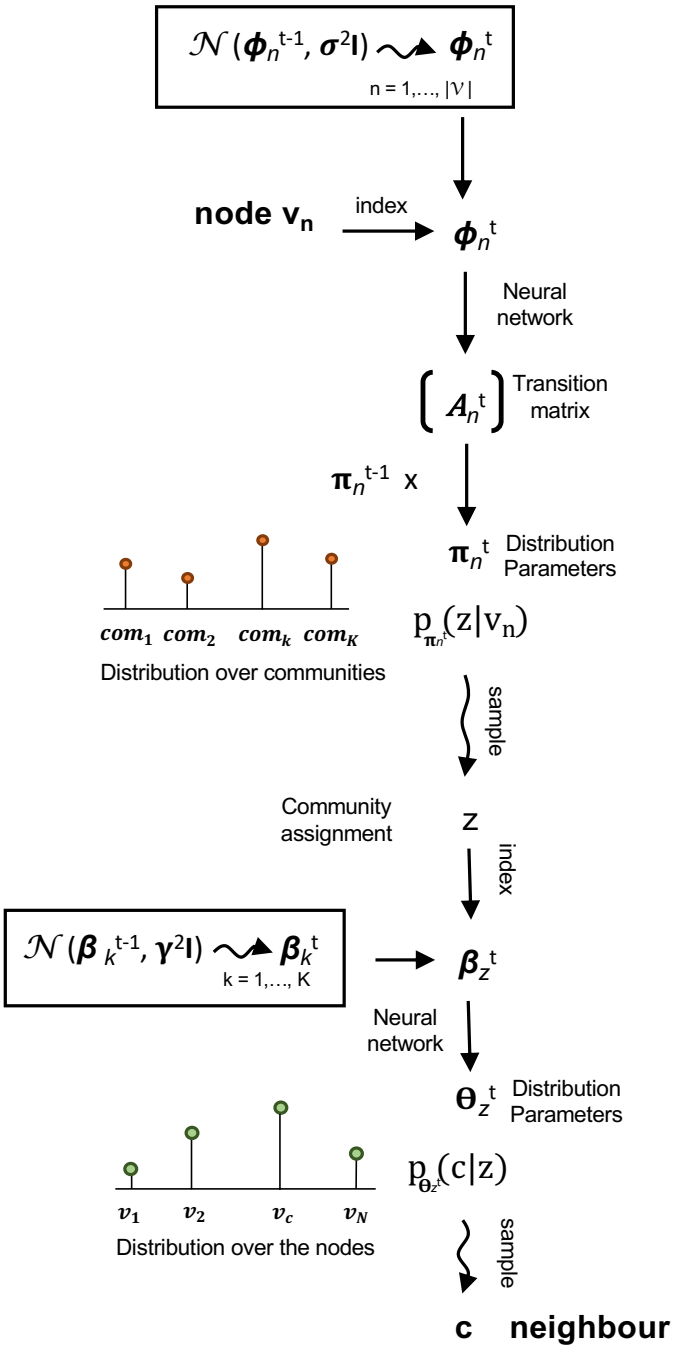


Figure 5.3: This figure presents GRADE's edge generative process for a source and target node pair at time step t. First, given a source node v_n, I sample its embedding, ϕ_n, from its prior distribution. Then, I transform this embedding into a community transition matrix, which I use to update the parameters of the multinomial distribution over the communities for node v_n. I sample a community assignment z from the community distribution of v_n. Then, I use the corresponding community embedding $\boldsymbol{\beta}_z^t$, and use it to parameterize a multinomial distribution over the nodes for community z after a neural network transformation. Finally, I sample a node neighbour c.

5.5.4 Inference Algorithm

In this chapter, I consider a dataset $\mathcal{D}$ comprising a set of node links $(v_n^t,\ c_{n,m}^t)$ for a sequence of graph snapshots $G^1, \ldots, G^{\mathrm{T}}$. In the proposed dynamic graph model, the latent variables are the hidden community assignments $z_{n,m}^t$, and the evolving node and community representations $\boldsymbol{\phi}_n^t$ and $\boldsymbol{\beta}_k^t$. The logarithm of the marginal probability of the observations is given by the sum of the log probability of each observed temporal edge $\log p(c_{n,m}^t|v_n^t)$ of all nodes in Γ:

$$\log p(\mathcal{D}) = \sum_{t=1}^{T} \sum_{n=1}^{|V^t|} \sum_{m=1}^{deg(v_n^t)} \log p_{\boldsymbol{\theta}^t, \boldsymbol{\pi}^t}(c_{n,m}^t|v_n^t) \tag{5.6}$$

Exact inference of the posterior $p(\boldsymbol{\phi}, \boldsymbol{\beta}, z|\mathcal{D})$ is intractable. Instead, I resort to variational methods as a means of approximation. Variational inference provides flexibility when choosing an appropriate family of distributions $q(\boldsymbol{\phi}, \boldsymbol{\beta}, z|\mathcal{D})$ as an approximation to the true posterior. Choosing such a family of variational distributions which is sufficiently flexible and adheres to the conditional dependencies of the hidden variables while allowing for an efficient optimization procedure is one of the main challenges in variational inference. The variational approximation I choose for the model takes the form:

$$\begin{aligned} q(\boldsymbol{\phi}, \boldsymbol{\beta}, z) = & \prod_{t=1}^{T} \prod_{n=1}^{|\mathcal{V}|} q(\boldsymbol{\phi}_n^t|\boldsymbol{\phi}_n^{1:t-1}) \\ & \times \prod_{t=1}^{T} \prod_{k=1}^{K} q(\boldsymbol{\beta}_k^t|\boldsymbol{\beta}_k^{1:t-1}) \\ & \times \prod_{t=1}^{T} \prod_{n=1}^{|V^t|} \prod_{m=1}^{deg(v_n^t)} q(z|v_n^t, c_{n,m}^t) \end{aligned} \tag{5.7}$$

The structure of the approximation has certain advantages which I discuss below. The goal of variational inference is to optimize the parameters of the approximated posterior by minimizing the Kullback-Leibler (KL) divergence between the true posterior and its approximation. This procedure is known as minimizing the evidence lower bound (ELBO) [110]:

$$\mathcal{L} = E_q[\log p(\mathcal{D}, \boldsymbol{\phi}, \boldsymbol{\beta}, z) - \log q(\boldsymbol{\phi}, \boldsymbol{\beta}, z)] \tag{5.8}$$

The ELBO for GRADE can be formulated as a neighbour reconstruction loss, and a sum of KL regularization terms between the priors and posteriors of each of the latent

variables. At each time step t, the ELBO from Eq.5.8 can be expressed as:

$$\mathcal{L}^t = E_{z^t \sim q(z|v^t,c^t)}[\log p_{\boldsymbol{\theta}^t}(c|z^t)] - KL[q(z|v^t,c^t)||p_{\boldsymbol{\pi}^t}(z|v^t)] - \\ \sum_k KL[q(\boldsymbol{\beta}_k^t|\boldsymbol{\beta}_k^{1:t-1})||p(\boldsymbol{\beta}_k^t|\boldsymbol{\beta}_k^{t-1})] - \frac{1}{|\mathcal{V}|}\sum_n^{|\mathcal{V}|} KL[q(\boldsymbol{\phi}_n^t|\boldsymbol{\phi}_n^{1:t-1})||p(\boldsymbol{\phi}_n^t|\boldsymbol{\phi}_n^{t-1})] \tag{5.9}$$

Note that I scale down the contribution of the KL divergence loss between the prior and posterior for the node representations by multiplying by $\frac{1}{|\mathcal{V}|}$, that is the size of the vertex set. I find that averaging over all nodes prevents from overpowering the other loss terms.

Algorithm 2 GRADE Inference Algorithm

Input: Edges $(v_n^t, c_{n,m}^t)$ in dynamic graph $\Gamma = G^1, \ldots, G^T$.
Initialize all generative and variational parameters.
Initialize $\boldsymbol{\beta}^0$, $\boldsymbol{\phi}^0$ as learnable parameters. $\boldsymbol{\pi}^0$ is uniform over K.
for iterations 1, 2, 3, ... **do**
  for graph G^t in $G^1, \ldots, G^T$ **do**
    ◇ **Sample community representations from posteriors**
    for k in $1, \ldots, K$ **do**
      $\boldsymbol{\beta}_k^t \sim q(\boldsymbol{\beta}_k^t|\boldsymbol{\beta}_k^{1:t-1})$
    end for
    ◇ **Sample node representation for complete vertex set from posteriors**
    for n in $1, \ldots, |\mathcal{V}|$ **do**
      $\boldsymbol{\phi}_n^t \sim q(\boldsymbol{\phi}_n^t|\boldsymbol{\phi}_n^{1:t-1})$
    end for
    ◇ **Sample community assignments for edges**
    for edges $(v_n^t, c_{n,m}^t)$ in G^t **do**
      $z_{n,m}^t \sim q(z|v_n^t, c_{n,m}^t)$
      $c_{n,m}^t \sim p_{\boldsymbol{\theta}_z^t}(c|z_{n,m}^t)$
    end for
  end for
  Estimate ELBO (eq. (5.9)) and update generative and variational parameters via backpropagation.
end for

Forming posteriors over the node and community embeddings. The variational distributions over the node and community representations depend on all of their respective historical states. I capture this temporal dependency with Gated Recurrent Units (GRU). Note that any other recurrent mechanism (e.g. Long short-term memory networks) can be used for this purpose, as well as temporal attention, such as in [193]. I model the outputs of both $q(\boldsymbol{\beta}_k^t|\boldsymbol{\beta}_k^{1:t-1})$ and $q(\boldsymbol{\phi}_n^t|\boldsymbol{\phi}_n^{1:t-1})$ as Gaussian distributions, similarly to their priors. In the posteriors however, the means and diagonal covariance vectors are given by the outputs of their respective GRU units. This approach in which some of the dependency of the hidden variables is retained, in this case over their historical states, is known as structured variational inference [94, 195]. There are several advantages to this approach: 1) it allows to easily infer the posterior distributions of the node and community representations at future time steps by running the GRUs after training; 2) I explicitly retain the historical dependence of the node and community embeddings, which increases the *fidelity of the approximation to the true posterior*; 3) the number of *model parameters introduced by the GRU module are negligible*. The alternative is using mean-field inference which would introduce extra parameters for each community and node at each time step. This would make the number of model parameters scale linearly with time. In my implementation, *the number of model parameters does not scale with time, but does scale with the number of nodes and communities*.

Inferring the community assignment hidden variable z. The difference between the approximated prior over the community assignments $p_{\boldsymbol{\pi}}(z|v)$ and the approximated posterior $q(z|v,c)$ is in the dependence on the neighbour c. In principle, this dependency can be easily integrated in the parametrization via amortized inference [68]. More specifically, I use both embeddings of the source, v, and target nodes, c, as inputs to the transformation ψ generating the community transition matrix. This *introduces no new model parameters* as the same ψ function can simply be reused. Also, the structure of the variational distribution over the assignments z enables an efficient procedure for inferring edge labels, as well as community memebership approximation as follows:

$$p(z|v) \approx \frac{1}{|N(v)|} \sum_{c \in N(v)} q(z|v,c) \tag{5.10}$$

where $N(v)$ is the set of neighbors of node v. The procedure is also applicable on future test graphs. Optimizing the lower bound (eq. (5.8)) w.r.t. all parameters is performed based on stochastic optimization using the reparametrization trick [111] and Gumbel-Softmax reparametrization [105, 139] to obtain gradients. Refer to Algorithm 2 for a summary of the inference procedure.

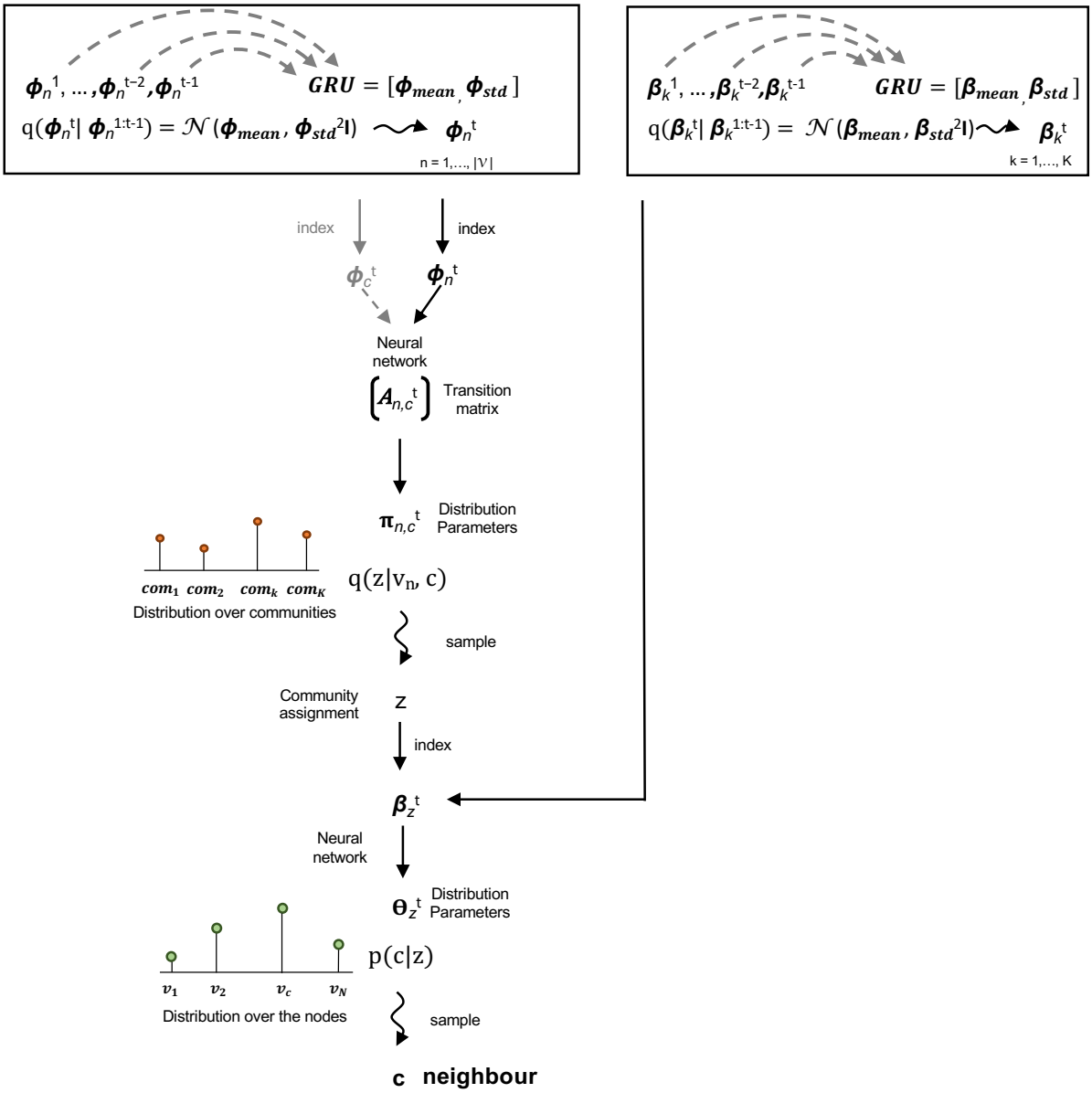


Figure 5.4: The inference algorithm I propose to learn the parameters of GRADE. The grey dashed lines denote variable dependencies that are necessary for the variational approximations of the intractable posteriors. The temporal dependency of the node and community embeddings is captured with Gated Recurrent Units (GRUs), which use all previous historical states to produce the mean and covariance vectors of their respective Gaussian variational approximations. In order to retain the dependency of the approximation over the communities, $q(z|v_n, c)$, on both the source and target nodes, I use both source and target node embeddings as inputs to the neural network which produces the community transition matrix.

Implementation. To implement GRADE, I use GRUs (one for the node and community evolution respectively) with a single layer. For the ψ and ζ functions, which are used to generate the parameters of the distributions over the communities and nodes respectively, I use simple linear layers. These are all the components required.

5.6 Experiments

I evaluate the proposed model on the tasks of dynamic link prediction, dynamic community detection and predicting community-scale dynamics against state-of-the-art baselines. Furthermore, I provide visualizations for a qualitative assessment.

5.6.1 Datasets

I use three discrete-time dynamic networks based on the DBLP, Wikipedia and Reddit datasets to evaluate the proposed method. A summary is provided in Table 5.2.

DBLP is a computer science bibliography which provides publication and author information from major journals and proceedings. I pre-process the DBLP dataset to identify the top 10,000 most prolific authors in terms of publication count in the years 2000-2018 inclusive. For each year, I construct a graph snapshot in which I connect author nodes if they have co-authored a publication in the same year. I stratify the DBLP dataset in 8 sub-fields, that is communities, based on publication venue. The communities comprise Artificial Intelligence, Computational Linguistics, Programming Languages, Data Mining, Databases, Systems, Hardware, Theory. I produce yearly labels for authors if over half of their annual publications fall within the same research category.

Reddit is a timestamped hyperlink network between subreddits [118]. An edge in the dataset represents a subreddit comment posting a hyperlink to another subreddit. Each edge has a timestamp associated with it. The complete dataset spans 2.5 years from January 2014 to April 2017 (overall 40 months). I divide the edges in 4-month blocks to create 10 graph snapshots spanning 40 months.

Wikipedia is a temporal network representing users editing each other's Talk pages [118, 170], and spans data between years 2002 and 2007 inclusive. An edge (v, c, t) means that user v edited user c's page at time t. During pre-processing, I first identify the 5,000 most active editors in the complete dataset. Then, I divide the edges for years 2006 and 2007 in 8-week periods (13 periods in total). The edges for each of the periods comprise a single graph snapshot.

Table 5.2: Datasets statistics. I use the proportion of time steps a node is present in to measure average node activity. The rate of context dynamics is captured by the Jaccard coefficient between the sets of immediate neighbours in consecutive time steps across all active nodes (lower coefficient suggests high rate of context dynamics).

	DBLP	**Reddit**	**Wikipedia**
# Nodes	10,000	35,776	5,000
# Links	374,911	180,662	482,069
Node Activity	0.47	0.25	0.76
Context Dynamics	0.30	0.24	0.13
Train snapshots	13	6	7
Val snapshots	3	2	3
Test snapshots	3	2	3
Label Rate	0.083	-	-

5.6.2 Baseline methods

I compare GRADE against five baselines comprising three static and two dynamic methods. The static methods are: **DeepWalk** [175], **node2vec** [78] and **vGraph** [215]. The dynamic graph methods consist of: **DynamicTriad** [255] and **DySAT** [193]

5.6.3 Evaluation Metrics

Dynamic link prediction. An important application of dynamic graph embedding is capturing the pattern of temporal edge formation in the training set to predict edges in future time steps. For all baselines I use a metric of similarity between node representations (Euclidean distance or dot product) as a predictor of connectivity, following each method's implementation. For static methods, I aggregate all observed edges in the training set in a single graph to produce node embeddings. For dynamic baselines, the vertex representations at the last training step are used. For GRADE, I train the model, and infer the posterior distributions of the node and community representations at the test time steps. I do this by using the trained GRUs to produce the node and community embeddings at the future time steps.

I evaluate dynamic link prediction performance using the metric *mean average rank* (MAR). To calculate mean average rank, I first produce a ranking of candidate neighbours,

spanning the complete vertex set, for each source node v^{test} in the test set edge list. Then, I identify the rank of the ground truth neighbour and average over all test edges. The ranking is produced via a similarity measure of the node embeddings for all baseline methods. For GRADE and vGraph, I use their generative models to produce a distribution over the neighbours for each source node v^{test} in the test set edge list. The procedure for GRADE is as follows: first, I use the trained GRUs to produce the node and community embeddings at test time step t^{test}. Then, I use the proposed generative process to produce the parameters of the distributions over the nodes and communities, that is $\theta^{t^{test}}$ and $\pi^{t^{test}}$. Finally, I can produce a distribution over the neighbours by summing over all possible community assignments as follows:

$$p(c|v^{test}) = \sum_{z} p_{\boldsymbol{\theta}^{t^{test}}}(c|z) p_{\boldsymbol{\pi}^{t^{test}}}(z|v^{test}) \tag{5.11}$$

I use the node probabilities to produce a ranking of candidate neighbours for source node v^{test}. The procedure for vGraph is very similar, the difference being that vGraph cannot generate node and community embeddings at future time steps. This allows to *directly assess the benefit of the temporal evolution* model in GRADE.

Dynamic community detection is another relevant use case for the proposed method. More specifically, I leverage historical information by training a model on the train time steps, and infer non-overlapping communities given the edges in the test set. For GRADE, this is achieved by inferring the hidden community assignment variable z. First, we use the trained GRUs to produce the node and community embeddings at the future time steps, and then we produce community assignments for nodes in the test set using Eq. 5.10. I evaluate performance on this task using *Normalized Mutual Information* (NMI) [227] and *Modularity*. Publicly available dynamic network datasets with labelled evolving communities are difficult to obtain. I use the DBLP dataset which I have manually labelled as described in subsection 5.3.1. Datasets.

Predicting community-scale dynamics is another novel application of GRADE. I demonstrate this capability by inferring the community representations (i.e., the posterior distribution over the nodes for each community) for the test time steps and producing rankings of the most probable nodes. A vertex predicted to have high probability for a given community should also be integral to its structure. I evaluate performance on this task by calculating Spearman's rank correlation coefficient between the predicted node probabilities of the top-250 vertices in community k and the same nodes' centrality as measured by the number of links to vertices assigned to the same community k on the test set.

5.6.4 Experimental Procedure

I cross-validate all methods and identify the best set of hyperparameters on the task of dynamic link prediction via grid search. The train/validation/test splits are done across time steps as shown in Table 5.2. I use no node attributes in any of the experiments and set the node embedding dimensionality to 128 for all methods. For all baselines with the exception of vGraph and GRADE, I apply K-means to the learnt vertex representations to identify non-overlapping communities. Further, since the majority of baseline methods (other than vGraph and GRADE) do not produce distributions over the nodes for each community, I use k-nearest neighbours algorithm to identify the top-250 nodes closest in representation space to each cluster's centroid for the task of predicting community-scale dynamics. For consistency between baselines, I determine the number of communities to be detected as part of the cross-validation procedure for GRADE. These were 8 communities for DBLP, 25 for Reddit and 50 for Wikipedia. The implementations provided by the authors are used for all baselines. I train GRADE using the Adam optimizer [109] with an initial learning rate of 0.005 which is decayed by 0.99 every 100 iterations. To save the best models for GRADE and vGraph, I train for 10,000 epochs on the task of dynamic link prediction and select the models with lowest mean average rank on the validation set. The same models were used in the evaluation for all tasks. I use procedures provided by the authors' implementations of DySAT and DynamicTriad to save best-performing models. Owing to the size of dynamic networks, full-batch training cannot be used. I train GRADE stochastically by splitting the edges at each time step in equally sized batches comprising $\sim 10,000 - 75,000$ edges in the experiments. Since the GRADE model is transductive we report results on nodes that have been observed in the training set. All results are averaged across 4 runs. Training GRADE on the proposed datasets requires up to ~ 12 hours on a 12GB NVIDIA TitanX GPU.

5.6.5 Results

Results on dynamic link prediction are shown in table 5.3. GRADE outperforms noticeably on all datasets. More specifically, for DBLP there is a 58% improvement in mean average rank compared to the second-best method, for Reddit - 51% improvement compared to second-best, and for Wikipedia - 70%. Often in real-world graphs the node-degree distribution is highly skewed and nodes cluster around hubs, which are densely connected. These hub nodes give rise to community structure in graphs [7]. As GRADE learns to correctly perform community assignment and attribute high probability to hub nodes, we expect GRADE to show favourable performance. These results demonstrate that *introducing an explicit inductive bias about the existence of community structure contributes to the task of dynamic link prediction.* Further, projecting the embeddings for

Table 5.3: Mean average rank (MAR) results on dynamic link prediction. Lower values are better. **Best** and second-best results are marked in bold and underlined respectively.

	DBLP	**Reddit**	**Wikipedia**
DeepWalk	$1,757 \pm 1$	$4,322 \pm 2$	$1,223 \pm 1.2$
Node2Vec	$2,418 \pm 9$	$6,863 \pm 28$	1,131 $\pm$ 2
DySAT	$1,505 \pm 0.1$	$4,040 \pm 5$	$1,546 \pm 1$
DynTriad	$1,905 \pm 12$	$1,575 \pm 36$	$1,390 \pm 13$
GRADE	**605 $\pm$ 39**	**601 $\pm$ 8**	**343 $\pm$ 6**
vGraph	1,436 $\pm$ 33	1,223 $\pm$ 33	$1,439 \pm 3$

future time steps, as opposed to re-using the ones learnt at the last training time step for dynamic prediction as other baselines do, can also be viewed as a performance advantage.

Results on dynamic community detection are presented in Table 5.4. In this task GRADE also outperforms all baselines on all datasets. Firstly, comparing against vGraph, the significance of injecting temporal evolution in the node and community embeddings via Gaussian random walks can be evaluated. The static vGraph cannot capture these dynamics, hence *relative improvement of up to* 24% *for GRADE is observed*, depending on the dataset. Secondly, all other baseline methods produce noticeably weaker results even than vGraph. This discrepancy in performance can be attributed to the efficient mechanism for community assignment (see Eq.5.10) for both GRADE and vGraph. In contrast, for other baselines K-Means clustering on the node embeddings has to be applied to identify communities, which proves to be not as effective as explicitly inferring the hidden community assignment variable.

Results on predicting community-scale dynamics. Similarly to other experiments, GRADE also outperforms on this task as can be seen in Table 5.5. More specifically, predicting the most structurally significant nodes for future time steps corresponds to correctly identifying the highly probable nodes for the communities. In GRADE, this can easily be accomplished by inferring the community embeddings for future time steps, producing the distributions over the nodes and taking the top 250 nodes. Compared to vGraph, which can also produce node distributions but does not capture time dynamics, GRADE performs better ($5\% - 10\%$ relative improvement). Compared to other baselines, which have to resort to applying the k-nearest neighbours algorithm on the produced node

Table 5.4: Modularity results on dynamic community detection.

	DBLP	**Reddit**	**Wikipedia**
DeepWalk	0.295 ± 0.002	0.146 ± 0.048	0.115 ± 0.001
Node2vec	0.314 ± 0.002	0.270 ± 0.005	0.113 ± 0.001
DySAT	0.306 ± 0.0	0.198 ± 0.005	0.071 ± 0.0
DynTriad	0.188 ± 0.001	0.072 ± 0.009	0.092 ± 0.001
GRADE	**0.383 ± 0.002**	**0.368 ± 0.004**	**0.139 ± 0.004**
vGraph	0.374 ± 0.001	0.296 ± 0.003	**0.138 ± 0.002**

embeddings to find the closest 250 nodes to each cluster centroid, GRADE outperforms substantially. Most notably, dynamic baselines DySAT and DynamicTriad are significantly inferior to GRADE's performance on all datasets.

Table 5.5: Correlation between node probability and node community centrality.

	DBLP	**Reddit**	**Wikipedia**
DeepWalk	-0.094 ± 0.060	0.241 ± 0.014	0.133 ± 0.016
Node2vec	0.151 ± 0.030	0.445 ± 0.016	0.094 ± 0.010
DySAT	0.053 ± 0.0	0.200 ± 0.005	0.080 ± 0.002
DynTriad	0.189 ± 0.038	0.124 ± 0.044	0.148 ± 0.019
GRADE	**0.323 ± 0.009**	**0.492 ± 0.019**	**0.289 ± 0.002**
vGraph	0.307 ± 0.004	0.448 ± 0.011	0.274 ± 0.010

Table 5.6: NMI results on DBLP.

	NMI
DeepWalk	0.401 ± 0.007
Node2vec	$\underline{0.4143 \pm 0.004}$
DySAT	$\underline{0.4137 \pm 0.0}$
DynTriad	0.103 ± 0.007
GRADE	$\mathbf{0.429 \pm 0.015}$
vGraph	0.368 ± 0.002

Finally, I also report results on **NMI on the DBLP dataset** in Table 5.6. Again, GRADE produces the best results followed by DySAT and node2vec.

In Figure 5.5, I visualize the temporal evolution of the top 10 most probable authors from a community strongly associated with Artificial Intelligence, learnt by GRADE on all time steps from the DBLP computer science bibliography. It can be observed that the top authors in each year work within the same general research area (*coherence*) and the community is broadly in agreement with historical events (*interpretability*). For example, GRADE assigns high probability to influential researchers like Yoshua Bengio and Ian J. Goodfellow in later years.

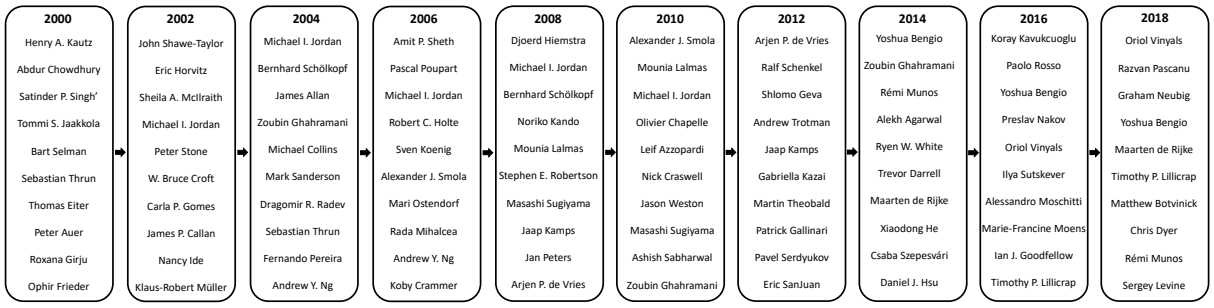


Figure 5.5: Temporal evolution of top-10 authors within a community broadly corresponding to Artificial Intelligence and learnt by GRADE on the DBLP dataset for years 2000-2018 inclusive.

5.7 Summary

In this chapter, I propose GRADE - a method which jointly learns evolving node and community representations in discrete-time dynamic graphs. I achieve this with an edge generative mechanism modelling the interaction between local and global graph structures via node and community multinomial distributions. These distributions are parameterized with neural network transformations of the learnt embeddings, and evolved over time with a Gaussian state-space model. Moreover, I introduce transition matrices to explicitly capture node dynamics. The proposed generative mechanism acts as an inductive bias and allows the model to exploit the inherent community structure of the data as well as implies smooth temporal evolution. I validate the effectiveness of GRADE on real-world datasets and observe significant performance gains against state-of-the-art baselines. In addition to incorporating prior knowledge about the edge-generative mechanism in dynamic graphs, I incorporate the principles of parameter-efficiency by minimizing the use of model parameters. More specifically, I avoid using mean-field inference and resort to amortized inference and use neural networks to produce distribution parameters. This makes the number of model parameters independent of the number of time steps.

Chapter 6

Conclusion and future work

The overarching goal of this thesis is to address the challenges of overparameterization and data efficient learning in deep neural networks. In brief, the contributions outlined in my dissertation achieve this goal either by proposing more parameter-efficient deep learning models, or by incorporating structural inductive biases, that is prior knowledge, about the machine learning task we are trying to solve. In the Conclusion, I summarize the contributions of each chapter once again, and then I propose several avenues for further research, which also utilize principles of efficiency as proposed in this document.

6.1 Summary of contributions

In **Chapter 3**, I extend parameter-efficient variants of the conventional convolutional operator, that is the separable convolution and grouped convolution, to 3D biomedical images, and provide an open-source implementation. Further, I study their applicability on both supervised and unsupervised neuroimaging tasks. First, on the task of Mild Cognitive Impairment to Alzheimer's disease prognostication, I achieve state-of-the-art results, as well as demonstrate that the proposed model learns to rely on brain areas already associated with the development of Alzheimer's for prediction. Second, I propose a multi-modal unsupervised framework for learning subject embeddings from neuroimage data via an image reconstruction loss. In both applications the result is a high-performing and parameter-efficient neural network which can be viably trained despite the high input dimensionality and multi modal inputs. Work associated with Chapter 3 is responsible for several publications [208, 204, 42, 43]. The idea of efficient multi-modal medical image integration was also adopted by an MPhil student I co-supervised (D. Taylor) who published his work in the *ML4H Workshop at NeurIPS* [222]. Finally, I also applied a similar methodology to predict brain age from MRI data for the PropagAgeing EU project.

In **Chapter 4**, I tackle a research problem which had been overlooked thus far, that is efficiently training deep neural network models on resource-constrained devices. Existing works had solely focused on efficient inference. In my thesis I propose a framework which starts by randomly sampling subnetworks within the full untrained neural network model, and runs them on batches of the input data, in order to associate a saliency metric with each model parameter *as well as* learn the parameters. I then use reinforcement learning and the observed salincies to identify the best performing subnetwork within the model and fine-tune the parameter values. As a result, the proposed method can simultaneously identify a compact and high-performing subnetwork (within a pre-defined memory utilization budget) and learn its weights. The approach consistently outperforms a random baseline. The method was published at the *British Machine Vision Conference (BMVC)* and the *EMC2 Workshop at NeurIPS* [206, 207]. I also supervised a Part III student who worked within the same area and published his thesis at the *Workshop on Automated Machine Learning at ICML* [70].

In **Chapter 5**, I tackle the problem of unsupervised dynamic graph representation learning by introducing GRADE. From a graph perspective, the novelty is that I propose a deep generative probabilistic model whereas other works use graph convolutional networks for the same task. The advantage of probabilistic modelling is that it allows us to inject prior knowledge about the problem, or a *structural inductive bias*, to improve data learning efficiency. In GRADE, this is achieved by using community assignment as a latent variable, hence acknowledging the community structure of real-world networks, and assuming smooth temporal evolution void of disruptive events. In terms of parameter-efficient implementation, I rely on amortized inference and parameter sharing for producing the parameters of the prior and posterior distributions. Furthermore, memory utilization is independent of the number of time steps in the data. The model outperforms state-of-the-art baselines on dynamic link prediction, dynamic community detection, and the novel task of predicting future structurally significant nodes. A manuscript describing the work is currently uder review at the *IEEE Transactions on Neural Networks and Learning Systems*.

6.2 Future research directions

Here, I briefly summarize several research directions which are tightly linked to parameter or data efficiency, and conceptually build upon the methods presented in this thesis. The goal of this section is not to be exhaustive but rather to present the research directions which I would hope to personally contribute to after the defence of my dissertation, although the suggested avenues are very diverse.

- *Distilling knowledge from multi-modal teacher networks to compact students for medical imaging applications.* Going beyond parameter-efficient architectures, we can simultaneously address the challenges of data collection and model size reduction for medical deep learning applications. For example, it has been shown that using Positron Emission Tomography (PET) imaging can significantly help in differentiating Alzheimer's disease from other types of dementia [141]. Similarly, the expression of apolipoprotein E (APOE), or APOe4, has been identified as a major risk factor for AD [108]. Clearly, augmenting a computer-aided diagnostic system with these input modalities would improve performance. The issue, however, is that PET imaging involves ionizing radiation and screening for APOe4 expression is an invasive procedure. To circumvent the need for patients to undergo such potentially dangerous procedures, it would be advantageous to train a highly multi-modal teacher network and distill its knowledge [91] to a unimodal student which utilizes only safe and cheap to collect patient data [80]. The teacher can learn the correlations between the various modalities and encode them in the compact model, hence simultaneously boosting performance and producing a compact architecture.

- *Applications of on-device training in federated learning and personalization.* The goal of this research direction is to explore use cases for efficient on-device training procedures, such as the one proposed in Chapter 4 of this dissertation. Two immediate application areas for such a framework are 1) federated learning and 2) personalization. Federated learning allows distributed edge devices to collaboratively learn a shared model while keeping the training data local [113]. A major bottleneck for federated learning is efficient on-device training [145]. The method from Chapter 4 could reduce the computational and memory burden to train on local data resulting in improved battery life and communicating fewer parameters to the global model. Further, we could modify a pre-trained local model to fit a specific scenario (personalization) by minimally changing the model's parameters based on user data without ever sharing it. Again, efficient on-device training would be required to achieve this. Recent work [125] attempts to achieve a similar goal by introducing early exits to deep neural networks, which allows to bypass some of the layers, thus reduce capacity in the absence of sufficient training data. Both of these use cases help remedy data privacy concerns by reducing the need to share user data with the cloud.

- *Modelling dynamic connectivity in fMRI data.* The brain can be represented as a temporal graph, where nodes are spatially distributed regions-of-interest (ROIs), that is brain areas. The edges can be determined by a measure of dynamic functional connectivity (dFC) applied to the fMRI data [37]. Emerging research suggests that temporal dynamics of ROI communities are useful biomarkers for understanding

brain function and dysfunction [35]. The method proposed in Chapter 5 (GRADE) is perfectly equipped to capture the connectivity between brain areas and jointly cluster them into communities while retaining temporal dynamics. The only extension necessary is introducing multi-graph learning and subject embeddings so as to learn from a set of fMRI sequences from many patients. Consequently, the community and node embeddings of an fMRI sequence for a given subject would be a function of this subject's embedding. The resulting unsupervised model can be used to discover novel subject groupings by clustering the subject embeddings.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, pages 37–48, 2013.

[3] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 242–252. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/allen-zhu19a.html`.

[4] Sitaram Asur, Srinivasan Parthasarathy, and Duygu Ucar. An event-based framework for characterizing the evolutionary behavior of interaction graphs. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(4):1–36, 2009.

[5] Filippo Baldacci, Simone Lista, Sid E O'Bryant, Roberto Ceravolo, Nicola Toschi, Harald Hampel, Alzheimer Precision Medicine Initiative, et al. Blood-based biomarker screening with agnostic biological definitions for an accurate diagnosis within the dimensional spectrum of neurodegenerative diseases. In *Biomarkers for Alzheimer's Disease Drug Development*, pages 139–155. Springer, 2018.

[6] Robert Bamler and Stephan Mandt. Dynamic word embeddings. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 380–389. PMLR, 06–11 Aug 2017.

[7] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[8] Deborah E Barnes and Kristine Yaffe. The projected effect of risk factor reduction on alzheimer's disease prevalence. *The Lancet Neurology*, 10(9):819–828, 2011.

[9] Federico Battiston, Giulia Cencetti, Iacopo Iacopini, Vito Latora, Maxime Lucas, Alice Patania, Jean-Gabriel Young, and Giovanni Petri. Networks beyond pairwise interactions: structure and dynamics, 2020.

[10] Iman Beheshti, Hasan Demirel, Hiroshi Matsuda, Alzheimer's Disease Neuroimaging Initiative, et al. Classification of alzheimer's disease and prediction of mild cognitive impairment-to-alzheimer's conversion from structural magnetic resource imaging using feature ranking and a genetic algorithm. *Computers in biology and medicine*, 83:109–119, 2017.

[11] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[12] Yash Bhalgat, Yizhe Zhang, Jamie Menjay Lin, and Fatih Porikli. Structured convolutions for efficient neural network design. *Advances in Neural Information Processing Systems*, 33:5553–5564, 2020.

[13] David M Blei and John D Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120, 2006.

[14] Heiko Braak and EVA Braak. Staging of alzheimer's disease-related neurofibrillary changes. *Neurobiology of aging*, 16(3):271–278, 1995.

[15] Heiko Braak and Eva Braak. Development of alzheimer-related neurofibrillary changes in the neocortex inversely recapitulates cortical myelogenesis. *Acta neuropathologica*, 92(2):197–201, 1996.

[16] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE transactions on knowledge and data engineering*, 20(2):172–188, 2007.

[17] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014.

[18] Ronald S Burt. Positions in networks. *Social forces*, 55(1):93–122, 1976.

[19] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 891–900, 2015.

[20] Ramon Casanova, Benjamin Wagner, Christopher T Whitlow, Jeff D Williamson, Sally A Shumaker, Joseph A Maldjian, and Mark A Espeland. High dimensional classification of structural mri alzheimer's disease data based on large scale regularization. *Frontiers in neuroinformatics*, 5:22, 2011.

[21] Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 554–560, 2006.

[22] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Michael Carbin, and Zhangyang Wang. The lottery tickets hypothesis for supervised and self-supervised pre-training in computer vision models, 2021.

[23] Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework, results and applications. In *International Conference on International Conference on Machine Learning - Volume 28*, pages I–151–I–159. JMLR.org, 2013.

[24] Zhourong Chen, Yang Li, Samy Bengio, and Si Si. Gaternet: Dynamic filter selection in convolutional neural network via a dedicated global gating network. *CoRR*, abs/1811.11205, 2018.

[25] Fu Cheng-Yang. pytorch-vgg-cifar10. https://github.com/chengyangfu/pytorch-vgg-cifar10, 2019.

[26] Yun Chi, Xiaodan Song, Dengyong Zhou, Koji Hino, and Belle L Tseng. Evolutionary spectral clustering by incorporating temporal smoothness. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 153–162, 2007.

[27] Ting-Wu Chin, Cha Zhang, and Diana Marculescu. Layer-compensated pruning for resource-constrained convolutional neural networks. *CoRR*, abs/1810.00518, 2018.

[28] Hongyoon Choi, Kyong Hwan Jin, Alzheimer's Disease Neuroimaging Initiative, et al. Predicting cognitive decline with deep learning of brain metabolism and amyloid imaging. *Behavioural brain research*, 344:103–109, 2018.

[29] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[30] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations,*

ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings, 2016. URL http://arxiv.org/abs/1511.07289.

[31] Taco Cohen and Max Welling. Group equivariant convolutional networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2990–2999, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/cohenc16.html.

[32] Taco S. Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. *CoRR*, abs/1801.10130, 2018. URL http://arxiv.org/abs/1801.10130.

[33] Matthieu Courbariaux and Yoshua Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *CoRR*, abs/1602.02830, 2016.

[34] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[35] Eswar Damaraju, Elena A Allen, Aysenil Belger, Judith M Ford, S McEwen, DH Mathalon, BA Mueller, GD Pearlson, SG Potkin, A Preda, et al. Dynamic functional connectivity analysis reveals transient states of dysconnectivity in schizophrenia. *NeuroImage: Clinical*, 5:298–308, 2014.

[36] Christos Davatzikos, Priyanka Bhatt, Leslie M Shaw, Kayhan N Batmanghelich, and John Q Trojanowski. Prediction of mci to ad conversion, via mri, csf biomarkers, and pattern classification. *Neurobiology of aging*, 32(12):2322–e19, 2011.

[37] Gustavo Deco, Adrián Ponce-Alvarez, Dante Mantini, Gian Luca Romani, Patric Hagmann, and Maurizio Corbetta. Resting-state functional connectivity emerges from structurally and dynamically shaped slow linear fluctuations. *Journal of Neuroscience*, 33(27):11239–11252, 2013.

[38] André Delacourte, Jean-Philippe David, Nicolas Sergeant, L Buee, A Wattez, P Vermersch, F Ghozali, C Fallet-Bianco, F Pasquier, F Lebert, et al. The biochemical pathway of neurofibrillary degeneration in aging and alzheimer's disease. *Neurology*, 52(6):1158–1158, 1999.

[39] Emily L. Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, pages 1269–1277, 2014.

[40] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[41] Adji B Dieng, Francisco JR Ruiz, and David M Blei. The dynamic embedded topic model. *arXiv preprint arXiv:1907.05545*, 2019.

[42] Giovanna Maria Dimitri, Simeon Spasov, Andrea Duggento, Luca Passamonti, Nicola Toschi, et al. Unsupervised stratification in neuroimaging through deep latent embeddings. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 1568–1571. IEEE, 2020.

[43] Giovanna Maria Dimitri, Simeon Spasov, Andrea Duggento, Luca Passamonti, Pietro Lio, and Nicola Toschi. Multimodal image fusion via deep generative models. *bioRxiv*, 2021.

[44] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pages 1675–1685. PMLR, 2019.

[45] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[46] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

[47] Shaker El-Sappagh, Jose M Alonso, SM Riazul Islam, Ahmad M Sultan, and Kyung Sup Kwak. A multilayer multimodal detection and prediction model based on explainable artificial intelligence for alzheimer's disease. *Scientific reports*, 11(1): 1–26, 2021.

[48] Peter Elias, Amiel Feinstein, and Claude Shannon. A note on the maximum flow through a network. *IRE Transactions on Information Theory*, 2(4):117–119, 1956.

[49] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning so(3) equivariant representations with spherical cnns. *CoRR*, 2017. URL `http://arxiv.org/abs/1711.06721`.

[50] Ernesto Estrada and Naomichi Hatano. Communicability in complex networks. *Physical Review E*, 77(3):036111, 2008.

[51] Ernesto Estrada and Naomichi Hatano. Communicability graph and community structures in complex networks. *Applied Mathematics and Computation*, 214(2): 500–511, 2009.

[52] Yong Fan, Dinggang Shen, Ruben C Gur, Raquel E Gur, and Christos Davatzikos. Compare: classification of morphological patterns using adaptive regional elements. *IEEE transactions on medical imaging*, 26(1):93–105, 2006.

[53] Cleusa P Ferri, Martin Prince, Carol Brayne, Henry Brodaty, Laura Fratiglioni, Mary Ganguli, Kathleen Hall, Kazuo Hasegawa, Hugh Hendrie, Yueqin Huang, et al. Global prevalence of dementia: a delphi consensus study. *The lancet*, 366(9503): 2112–2117, 2005.

[54] Stephen E Fienberg and Stanley S Wasserman. Categorical data analysis of single sociometric relations. *Sociological methodology*, 12:156–192, 1981.

[55] Mikhail Figurnov, Aizhan Ibraimova, Dmitry P Vetrov, and Pushmeet Kohli. Perforatedcnns: Acceleration through elimination of redundant convolutions. In *Advances in Neural Information Processing Systems*, pages 947–955, 2016.

[56] Roman Filipovych, Christos Davatzikos, Alzheimer's Disease Neuroimaging Initiative, et al. Semi-supervised pattern classification of medical images: application to mild cognitive impairment (mci). *NeuroImage*, 55(3):1109–1119, 2011.

[57] Gary William Flake, Steve Lawrence, and C Lee Giles. Efficient identification of web communities. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–160, 2000.

[58] Gary William Flake, Steve Lawrence, C Lee Giles, and Frans M Coetzee. Self-organization and identification of web communities. *Computer*, 35(3):66–70, 2002.

[59] Lester Randolph Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8:399–404, 1956.

[60] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

[61] Santo Fortunato and Marc Barthelemy. Resolution limit in community detection. *Proceedings of the national academy of sciences*, 104(1):36–41, 2007.

[62] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics reports*, 659:1–44, 2016.

[63] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on knowledge and data engineering*, 19(3):355–369, 2007.

[64] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

[65] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Stabilizing the lottery ticket hypothesis. *arXiv preprint arXiv:1903.01611*, 2019.

[66] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3259–3269. PMLR, 13–18 Jul 2020.

[67] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Stabilizing the lottery ticket hypothesis, 2020.

[68] Samuel Gershman and Noah Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the annual meeting of the cognitive science society*, volume 36, 2014.

[69] Zoubin Ghahramani. An introduction to hidden markov models and bayesian networks. In *Hidden Markov models: applications in computer vision*, pages 9–41. World Scientific, 2001.

[70] Samuel Glass, Simeon Spasov, and Pietro Liò. Riccinets: Curvature-guided pruning of high-performance neural networks using ricci flow. *arXiv preprint arXiv:2007.04216*, 2020.

[71] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[72] Andrew V Goldberg and Robert E Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM (JACM)*, 35(4):921–940, 1988.

[73] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[74] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[75] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. Dyngem: Deep embedding method for dynamic graphs. *arXiv preprint arXiv:1805.11273*, 2018.

[76] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems*, 187:104816, 2020.

[77] Derek Greene, Donal Doyle, and Padraig Cunningham. Tracking the evolution of communities in dynamic social networks. In *2010 international conference on advances in social networks analysis and mining*, pages 176–183. IEEE, 2010.

[78] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[79] Sergio Grueso and Raquel Viejo-Sobera. Machine learning methods for predicting progression from mild cognitive impairment to alzheimer's disease dementia: a systematic review. *Alzheimer's research & therapy*, 13(1):1–29, 2021.

[80] Hao Guan, Chaoyue Wang, and Dacheng Tao. Mri-based alzheimer's disease prediction via distilling the knowledge in multi-modal data. *arXiv preprint arXiv:2104.03618*, 2021.

[81] Roger Guimera, Marta Sales-Pardo, and Luís A Nunes Amaral. Modularity from fluctuations in random graphs and complex networks. *Physical Review E*, 70(2): 025101, 2004.

[82] Yubraj Gupta, Ramesh Kumar Lama, Goo-Rak Kwon, Michael W Weiner, Paul Aisen, Michael Weiner, Ronald Petersen, Clifford R Jack Jr, William Jagust, John Q Trojanowki, et al. Prediction and classification of alzheimer's disease based on combined features from apolipoprotein-e genotype, cerebrospinal fluid, mr, and fdg-pet imaging biomarkers. *Frontiers in computational neuroscience*, 13:72, 2019.

[83] Amirhossein Habibian, Davide Abati, Taco Cohen, and Babak Ehteshami Bejnordi. Skip-convolutions for efficient video processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[84] Alexander Hammers, Richard Allom, Matthias J Koepp, Samantha L Free, Ralph Myers, Louis Lemieux, Tejal N Mitchell, David J Brooks, and John S Duncan. Three-dimensional maximum probability atlas of the human brain, with particular reference to the temporal lobe. *Human brain mapping*, 19(4):224–247, 2003.

[85] Harald Hampel, Nicola Toschi, Filippo Baldacci, Henrik Zetterberg, Kaj Blennow, Ingo Kilimann, Stefan J Teipel, Enrica Cavedo, Antonio Melo dos Santos, Stéphane Epelbaum, et al. Alzheimer's disease biomarker-guided diagnostic workflow using the added value of six combined cerebrospinal fluid candidates: Aβ1–42, total-tau, phosphorylated-tau, nfl, neurogranin, and ykl-40. *Alzheimer's & Dementia*, 14(4): 492–501, 2018.

[86] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[87] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. EIE: efficient inference engine on compressed deep neural network. In *ISCA*, pages 243–254. IEEE Computer Society, 2016.

[88] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[89] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 770–778, 2016.

[90] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[91] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[92] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *Advances in Neural Information Processing Systems, Deep Learning and Representation Learning Workshop*, 2015.

[93] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *arXiv preprint arXiv:2102.00554*, 2021.

[94] Matthew D Hoffman and David M Blei. Structured stochastic variational inference. In *Artificial Intelligence and Statistics*, 2015.

[95] Seyed Hani Hojjati, Ata Ebrahimzadeh, Ali Khazaee, Abbas Babajani-Feremi, Alzheimer's Disease Neuroimaging Initiative, et al. Predicting conversion from mci

to ad using resting-state fmri, graph theoretical approach and svm. *Journal of neuroscience methods*, 282:69–80, 2017.

[96] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

[97] John Hopcroft, Omar Khan, Brian Kulis, and Bart Selman. Tracking evolving communities in large linked networks. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5249–5253, 2004.

[98] Ehsan Hosseini-Asl, Robert Keynton, and Ayman El-Baz. Alzheimer's disease diagnostics by adaptation of 3d convolutional network. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 126–130. IEEE, 2016.

[99] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[100] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 2261–2269, 2017.

[101] David H Hubel and Torsten N Wiesel. 8. receptive fields of single neurones in the cat's striate cortex. In *Brain Physiology and Psychology*, pages 129–150. University of California Press, 2020.

[102] Yani Ioannou, Duncan Robertson, Roberto Cipolla, and Antonio Criminisi. Deep roots: Improving cnn efficiency with hierarchical filter groups. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1231–1240, 2017.

[103] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[104] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

[105] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[106] Georgios A Kaissis, Marcus R Makowski, Daniel Rückert, and Rickmer F Braren. Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence*, 2(6):305–311, 2020.

[107] Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307, 1970.

[108] Jungsu Kim, Jacob M Basak, and David M Holtzman. The role of apolipoprotein e in alzheimer's disease. *Neuron*, 63(3):287–303, 2009.

[109] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[110] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[111] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589, 2014.

[112] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[113] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[114] Igor O Korolev, Laura L Symonds, Andrea C Bozoki, and Alzheimer's Disease Neuroimaging Initiative. Predicting progression from mild cognitive impairment to alzheimer's dementia using clinical, mri, and plasma biomarkers via probabilistic pattern classification. *PloS one*, 11(2):e0138866, 2016.

[115] Gueorgi Kossinets and Duncan J Watts. Empirical analysis of an evolving social network. *science*, 311(5757):88–90, 2006.

[116] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[117] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

[118] Srijan Kumar, William L Hamilton, Jure Leskovec, and Dan Jurafsky. Community interaction and conflict on the web. In *Proceedings of the 2018 World Wide Web Conference*, pages 933–943, 2018.

[119] Andrea Lancichinetti and Santo Fortunato. Consensus clustering in complex networks. *Scientific reports*, 2(1):1–7, 2012.

[120] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014.

[121] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan V. Oseledets, and Victor S. Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. In *International Conference for Learning Representations*, 2015.

[122] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2:598–605, 1989.

[123] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999.

[124] Eunho Lee, Jun-Sik Choi, Minjeong Kim, Heung-Il Suk, Alzheimer's Disease Neuroimaging Initiative, et al. Toward an interpretable alzheimer's disease diagnostic model with regional abnormality representation via deep learning. *NeuroImage*, 202: 116113, 2019.

[125] Ilias Leontiadis, Stefanos Laskaridis, Stylianos I Venieris, and Nicholas D Lane. It's always personal: Using early exits for efficient on-device cnn personalisation. In *Proceedings of the 22nd International Workshop on Mobile Computing Systems and Applications*, pages 15–21, 2021.

[126] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *International Conference for Learning Representations*, 2016.

[127] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. Attributed network embedding for learning in a dynamic environment. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 387–396, 2017.

[128] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *Advances in neural information processing systems*, pages 2181–2191, 2017.

[129] Yu-Ru Lin, Yun Chi, Shenghuo Zhu, Hari Sundaram, and Belle L Tseng. Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. In *Proceedings of the 17th international conference on World Wide Web*, pages 685–694, 2008.

[130] Fanzhen Liu, Shan Xue, Jia Wu, Chuan Zhou, Wenbin Hu, Cecile Paris, Surya Nepal, Jian Yang, and Philip S Yu. Deep learning for community detection: progress, challenges and opportunities. *arXiv preprint arXiv:2005.08225*, 2020.

[131] Ke Liu, Kewei Chen, Li Yao, and Xiaojuan Guo. Prediction of mild cognitive impairment conversion using a combination of independent component analysis and the cox model. *Frontiers in human neuroscience*, 11:33, 2017.

[132] Lanlan Liu and Jia Deng. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. In *AAAI Conference on Artificial Intelligence*, pages 3675–3682. AAAI Press, 2018.

[133] Manhua Liu, Danni Cheng, Kundong Wang, Yaping Wang, Alzheimer's Disease Neuroimaging Initiative, et al. Multi-modality cascaded convolutional neural networks for alzheimer's disease diagnosis. *Neuroinformatics*, 16(3-4):295–308, 2018.

[134] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. *2017 IEEE International Conference on Computer Vision*, pages 2755–2763, 2017.

[135] Francois Lorrain and Harrison C White. Structural equivalence of individuals in social networks. *The Journal of mathematical sociology*, 1(1):49–80, 1971.

[136] Donghuan Lu, Karteek Popuri, Gavin Weiguang Ding, Rakesh Balachandar, and Mirza Faisal Beg. Multimodal and multiscale deep neural networks for the early diagnosis of alzheimer's disease using structural mr and fdg-pet images. *Scientific reports*, 8(1):1–13, 2018.

[137] Donghuan Lu, Karteek Popuri, Gavin Weiguang Ding, Rakesh Balachandar, Mirza Faisal Beg, Alzheimer's Disease Neuroimaging Initiative, et al. Multiscale deep neural network based analysis of fdg-pet images for the early diagnosis of alzheimer's disease. *Medical image analysis*, 46:26–34, 2018.

[138] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[139] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[140] Carlo Maj, Tiago Azevedo, Valentina Giansanti, Oleg Borisov, Giovanna Maria Dimitri, Simeon Spasov, Pietro Lio, and Ivan Merelli. Integration of machine learning

methods to dissect genetically imputed transcriptomic profiles in alzheimer's disease. *Frontiers in genetics*, 10:726, 2019.

[141] Charles Marcus, Esther Mena, and Rathan M Subramaniam. Brain pet in the diagnosis of alzheimer's disease. *Clinical nuclear medicine*, 39(10):e413, 2014.

[142] Kenneth Marek, Danna Jennings, Shirley Lasch, Andrew Siderowf, Caroline Tanner, Tanya Simuni, Chris Coffey, Karl Kieburtz, Emily Flagg, Sohini Chowdhury, et al. The parkinson progression marker initiative (ppmi). *Progress in neurobiology*, 95(4): 629–635, 2011.

[143] William R Markesbery. Neuropathologic alterations in mild cognitive impairment: a review. *Journal of Alzheimer's Disease*, 19(1):221–228, 2010.

[144] Kaspar Märtens and Christopher Yau. Basisvae: Translation-invariant feature-level clustering with variational autoencoders. In *International Conference on Artificial Intelligence and Statistics*, pages 2928–2937. PMLR, 2020.

[145] Akhil Mathur, Daniel J Beutel, Pedro Porto Buarque de Gusmão, Javier Fernandez-Marques, Taner Topal, Xinchi Qiu, Titouan Parcollet, Yan Gao, and Nicholas D Lane. On-device federated learning with flower. *arXiv preprint arXiv:2104.03042*, 2021.

[146] Niklas Mattsson, Henrik Zetterberg, Oskar Hansson, Niels Andreasen, Lucilla Parnetti, Michael Jonsson, Sanna-Kaisa Herukka, Wiesje M van der Flier, Marinus A Blankenstein, Michael Ewers, et al. Csf biomarkers and incipient alzheimer disease in patients with mild cognitive impairment. *Jama*, 302(4):385–393, 2009.

[147] Scott Mayer McKinney, Marcin Sieniek, Varun Godbole, Jonathan Godwin, Natasha Antropova, Hutan Ashrafian, Trevor Back, Mary Chesus, Greg S Corrado, Ara Darzi, et al. International evaluation of an ai system for breast cancer screening. *Nature*, 577(7788):89–94, 2020.

[148] AJ Mitchell and M Shiri-Feshki. Temporal trends in the long term risk of progression of mild cognitive impairment: a pooled analysis. *Journal of Neurology, Neurosurgery & Psychiatry*, 79(12):1386–1391, 2008.

[149] Tom M Mitchell. *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research ..., 1980.

[150] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[151] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[152] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[153] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. In *International Conference for Learning Representations*, 2017.

[154] Elaheh Moradi, Antonietta Pepe, Christian Gaser, Heikki Huttunen, Jussi Tohka, Alzheimer's Disease Neuroimaging Initiative, et al. Machine learning framework for early mri-based alzheimer's conversion prediction in mci subjects. *Neuroimage*, 104: 398–412, 2015.

[155] JC Morris, M Storandt, DW McKeel, EH Rubin, JL Price, EA Grant, and L Berg. Cerebral amyloid deposition and diffuse plaques in"normal"aging: Evidence for presymptomatic and very mild alzheimer's disease. *Neurology*, 46(3):707–719, 1996.

[156] L Mosconi, D Perani, S Sorbi, Karl Herholz, B Nacmias, Vjera Holthoff, Eric Salmon, J-C Baron, MTR De Cristofaro, A Padovani, et al. Mci conversion to dementia and the apoe genotype: a prediction study with fdg-pet. *Neurology*, 63(12):2332–2340, 2004.

[157] Lisa Mosconi, Miroslaw Brys, Lidia Glodzik-Sobanska, Susan De Santi, Henry Rusinek, and Mony J. de Leon. Early detection of alzheimer's disease using neuroimaging. *Experimental Gerontology*, 42(1-2):129–138, January 2007. doi: 10.1016/j.exger.2006.05.016. URL `https://doi.org/10.1016/j.exger.2006.05.016`.

[158] Susanne G Mueller, Michael W Weiner, Leon J Thal, Ronald C Petersen, Clifford R Jack, William Jagust, John Q Trojanowski, Arthur W Toga, and Laurel Beckett. Ways toward an early diagnosis in alzheimer's disease: the alzheimer's disease neuroimaging initiative (adni). *Alzheimer's & Dementia*, 1(1):55–66, 2005.

[159] Yu Nesterov. A method of solving a convex programming problem with convergence rate o (1/kˆ 2) o (1/k2). In *Sov. Math. Dokl*, volume 27, 1987.

[160] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003.

[161] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[162] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[163] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks, 2018.

[164] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14, 2001.

[165] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference 2018*, pages 969–976, 2018.

[166] Minh Hoai Nguyen and Fernando de la Torre. Optimal feature selection for support vector machines. *Pattern Recognition*, 43(3):584–591, March 2010. doi: 10.1016/j.patcog.2009.09.003. URL `https://doi.org/10.1016/j.patcog.2009.09.003`.

[167] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*, 2017.

[168] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[169] Gergely Palla, Albert-László Barabási, and Tamás Vicsek. Quantifying social group evolution. *Nature*, 446(7136):664–667, 2007.

[170] Ashwin Paranjape, Austin R Benson, and Jure Leskovec. Motifs in temporal networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 601–610, 2017.

[171] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, and Charles E Leisersen. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. *arXiv preprint arXiv:1902.10191*, 2019.

[172] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[173] Adrien Payan and Giovanni Montana. Predicting alzheimer's disease: a neuroimaging study with 3d convolutional neural networks. *arXiv preprint arXiv:1502.02506*, 2015.

[174] Judea Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proceedings of the 7th Conference of the Cognitive Science Society, University of California, Irvine, CA, USA*, pages 15–17, 1985.

[175] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[176] Chiara Pirazzini, Tiago Azevedo, Luca Baldelli, Anna Bartoletti-Stella, Giovanna Calandra-Buonaura, Alessandra Dal Molin, Giovanna Maria Dimitri, Ivan Doykov, Pilar Gómez-Garre, Sara Hägg, et al. A geroscience approach for parkinson's disease: conceptual framework and design of propag-ageing project. *Mechanisms of Ageing and Development*, page 111426, 2020.

[177] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

[178] Sandra Pusil, Stavros I Dimitriadis, María Eugenia López, Ernesto Pereda, and Fernando Maestú. Aberrant meg multi-frequency phase temporal synchronization predicts conversion from mild cognitive impairment-to-alzheimer's disease. *NeuroImage: Clinical*, 24:101972, 2019.

[179] Ubaldo Ramon-Julvez, Monica Hernandez, Elvira Mayordomo, et al. Analysis of the influence of diffeomorphic normalization in the prediction of stable vs progressive mci conversion with convolutional neural networks. In *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pages 1120–1124. IEEE, 2020.

[180] C Radhakrishna Rao. Information and the accuracy attainable in the estimation of statistical parameters. *Reson. J. Sci. Educ*, 20:78–90, 1945.

[181] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV (4)*, volume 9908 of *Lecture Notes in Computer Science*, pages 525–542. Springer, 2016.

[182] Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=S1gSj0NKvB`.

[183] Matthias Riemenschneider, Nicola Lautenschlager, S Wagenpfeil, J Diehl, A Drzezga, and A Kurz. Cerebrospinal fluid tau and β-amyloid 42 proteins identify alzheimer

disease in subjects with mild cognitive impairment. *Archives of Neurology*, 59(11): 1729–1734, 2002.

[184] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *International Conference for Learning Representations*, 2015.

[185] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[186] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

[187] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the national academy of sciences*, 105 (4):1118–1123, 2008.

[188] Maja Rudolph and David Blei. Dynamic embeddings for language evolution. In *Proceedings of the 2018 World Wide Web Conference*, pages 1003–1011, 2018.

[189] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[190] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, April 2015. doi: 10.1007/s11263-015-0816-y. URL `https://doi.org/10.1007/s11263-015-0816-y`.

[191] Marco Saerens, Francois Fouss, Luh Yen, and Pierre Dupont. The principal components analysis of a graph, and its relationships to spectral clustering. In *European conference on machine learning*, pages 371–383. Springer, 2004.

[192] Itay Safran and Ohad Shamir. Depth-width tradeoffs in approximating natural functions with neural networks. In *International Conference on Machine Learning*, pages 2979–2987. PMLR, 2017.

[193] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 519–527, 2020.

[194] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *arXiv preprint arXiv:1805.11604*, 2018.

[195] Lawrence K Saul and Michael I Jordan. Exploiting tractable substructures in intractable networks. In *Advances in neural information processing systems*, pages 486–492, 1996.

[196] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing*, pages 362–373. Springer, 2018.

[197] A. Serrano-Pozo, M. P. Frosch, E. Masliah, and B. T. Hyman. Neuropathological alterations in alzheimer disease. *Cold Spring Harbor Perspectives in Medicine*, 1 (1):a006189–a006189, September 2011. doi: 10.1101/cshperspect.a006189. URL `https://doi.org/10.1101/cshperspect.a006189`.

[198] Jennifer L. Shaffer, Jeffrey R. Petrella, Forrest C. Sheldon, Kingshuk Roy Choudhury, Vince D. Calhoun, R. Edward Coleman, and P. Murali Doraiswamy and. Predicting cognitive decline in subjects at risk for alzheimer disease by using combined cerebrospinal fluid, MR imaging, and PET biomarkers. *Radiology*, 266(2):583–591, February 2013. doi: 10.1148/radiol.12120010. URL `https://doi.org/10.1148/radiol.12120010`.

[199] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[200] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[201] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations, Conference Track Proceedings*, 2015.

[202] Tom AB Snijders and Krzysztof Nowicki. Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of classification*, 14(1): 75–100, 1997.

[203] Joshua A. Sonnen, Kathleen S. Montine, Joseph F. Quinn, John C.S. Breitner, and Thomas J. Montine. Cerebrospinal fluid biomarkers in mild cognitive impairment and dementia. *Journal of Alzheimer's Disease*, 19(1):301–309, January 2010. ISSN

18758908, 13872877. doi: 10.3233/JAD-2010-1236. URL https://doi.org/10.3233/JAD-2010-1236.

[204] Simeon Spasov, Luca Passamonti, Andrea Duggento, Pietro Liò, Nicola Toschi, Alzheimer's Disease Neuroimaging Initiative, et al. A parameter-efficient deep learning approach to predict conversion from mild cognitive impairment to alzheimer's disease. *Neuroimage*, 189:276–287, 2019.

[205] Simeon Spasov, Alessandro Di Stefano, Pietro Liò, and Jian Tang. Grade: Graph dynamic embedding. *arXiv preprint arXiv:2007.08060*, 2020.

[206] Simeon E Spasov and Pietro Liò. Dynamic neural network channel execution for efficient training. *British Machine Vision Conference (BMVC)*, 2019.

[207] Simeon E Spasov and Pietro Liò. Dynamic channel execution: on-device learning method for finding compact networks. *Workshop on Energy Efficient Machine Learning (EMC2), NeurIPS*, 2019.

[208] Simeon E Spasov, Luca Passamonti, Andrea Duggento, Pietro Liò, and Nicola Toschi. A multi-modal convolutional neural network framework for the prediction of alzheimer's disease. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1271–1274. IEEE, 2018.

[209] Suraj Srinivas and R. Venkatesh Babu. Data-free parameter pruning for deep neural networks. In *Proceedings of the British Machine Vision Conference*, pages 31.1–31.12, 2015.

[210] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[211] Jiahao Su, Jingling Li, Bobby Bhattacharjee, and Furong Huang. Tensorized spectrum preserving compression for neural networks. *arXiv preprint arXiv:1805.10352*, 2018.

[212] Yingcheng Su, Shunfeng Zhou, Yichao Wu, Xuebo Liu, Tian Su, Ding Liang, and Junjie Yan. Context-aware dynamic block. *CoRR*, abs/1902.10949, 2019.

[213] Peter R Suaris and Gershon Kedem. An algorithm for quadrisection and its application to standard cell placement. *IEEE Transactions on Circuits and Systems*, 35(3): 294–303, 1988.

[214] Cathie Sudlow, John Gallacher, Naomi Allen, Valerie Beral, Paul Burton, John Danesh, Paul Downey, Paul Elliott, Jane Green, Martin Landray, et al. Uk biobank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age. *Plos med*, 12(3):e1001779, 2015.

[215] Fan-Yun Sun, Meng Qu, Jordan Hoffmann, Chin-Wei Huang, and Jian Tang. vgraph: A generative model for joint community detection and node representation learning. In *Advances in Neural Information Processing Systems*, pages 512–522, 2019.

[216] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019.

[217] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

[218] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*, 2014.

[219] Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, et al. Convolutional neural networks with low-rank regularization. *arXiv preprint arXiv:1511.06067*, 2015.

[220] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[221] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.

[222] Devin Taylor, Simeon Spasov, and Pietro Liò. Co-attentive cross-modal deep learning for medical evidence synthesis and decision making. *arXiv preprint arXiv:1909.06442*, 2019.

[223] Jason R Taylor, Nitin Williams, Rhodri Cusack, Tibor Auer, Meredith A Shafto, Marie Dixon, Lorraine K Tyler, Richard N Henson, et al. The cambridge centre for ageing and neuroscience (cam-can) data repository: Structural and functional mri, meg, and cognitive data from a cross-sectional adult lifespan sample. *Neuroimage*, 144:262–269, 2017.

[224] Stefan J. Teipel, Enrica Cavedo, Simone Lista, Marie-Odile Habert, Marie-Claude Potier, Michel J. Grothe, Stephane Epelbaum, Luisa Sambati, Geoffroy Gagliardi, Nicola Toschi, Michael D. Greicius, Bruno Dubois, Harald Hampel, and and. Effect of

alzheimer's disease risk and protective factors on cognitive trajectories in subjective memory complainers: An INSIGHT-preAD study. *Alzheimer's & Dementia*, 14(9): 1126–1136, May 2018. doi: 10.1016/j.jalz.2018.04.004. URL `https://doi.org/10.1016/j.jalz.2018.04.004`.

[225] Matus Telgarsky. Benefits of depth in neural networks. In *Conference on learning theory*, pages 1517–1539. PMLR, 2016.

[226] Lucas Theis, Iryna Korshunova, Alykhan Tejani, and Ferenc Huszár. Faster gaze prediction with dense networks and fisher pruning. *CoRR*, abs/1801.05787, 2018.

[227] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. Learning deep representations for graph clustering. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[228] Geoffrey Hinton Tijmen Tieleman. rmsprop: Divide the gradient by a running average of its recent magnitude. *Neural Networks for Machine Learning. COURSERA*, 2012.

[229] Tong Tong, Qinquan Gao, Ricardo Guerrero, Christian Ledig, Liang Chen, Daniel Rueckert, and Alzheimer's Disease Neuroimaging Initiative. A novel grading biomarker for the prediction of conversion from mild cognitive impairment to alzheimer's disease. *IEEE Transactions on Biomedical Engineering*, 64(1):155–165, January 2017. doi: 10.1109/tbme.2016.2549363. URL `https://doi.org/10.1109/tbme.2016.2549363`.

[230] David C Van Essen, Stephen M Smith, Deanna M Barch, Timothy EJ Behrens, Essa Yacoub, Kamil Ugurbil, Wu-Minn HCP Consortium, et al. The wu-minn human connectome project: an overview. *Neuroimage*, 80:62–79, 2013.

[231] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[232] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *2018 European Conference on Computer Vision*, 2018.

[233] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

[234] Diego Vidaurre, Stephen M Smith, and Mark W Woolrich. Brain network dynamics are hierarchically organized in time. *Proceedings of the National Academy of Sciences*, 114(48):12827–12832, 2017.

[235] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[236] Huan Wang, Qiming Zhang, Yuehai Wang, and Haoji Hu. Structured probabilistic pruning for convolutional neural network acceleration. In *British Machine Vision Conference 2018*, page 149, 2018.

[237] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3349–3364, 2020.

[238] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E. Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *European Conference on Computer Vision*, pages 420–436, 2018.

[239] Stanley Wasserman, Katherine Faust, et al. *Social network analysis: Methods and applications*. Cambridge university press, 1994.

[240] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2074–2082, 2016.

[241] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[242] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S. Davis, Kristen Grauman, and Rogério Schmidt Feris. Blockdrop: Dynamic inference paths in residual networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8817–8826, 2018.

[243] Tianbao Yang, Yun Chi, Shenghuo Zhu, Yihong Gong, and Rong Jin. Detecting communities and their evolutions in dynamic social networks—a bayesian approach. *Machine learning*, 82(2):157–189, 2011.

[244] Yinchong Yang, Denis Krompass, and Volker Tresp. Tensor-train recurrent neural networks for video classification. In *International Conference on Machine Learning*, pages 3891–3900. PMLR, 2017.

[245] Luh Yen, Francois Fouss, Christine Decaestecker, Pascal Francq, and Marco Saerens. Graph nodes clustering based on the commute-time kernel. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 1037–1045. Springer, 2007.

[246] Luh Yen, Francois Fouss, Christine Decaestecker, Pascal Francq, and Marco Saerens. Graph nodes clustering with the sigmoid commute-time kernel: A comparative study. *Data & Knowledge Engineering*, 68(3):338–361, 2009.

[247] Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Yingyan Lin, Zhangyang Wang, and Richard G. Baraniuk. Drawing early-bird tickets: Toward more efficient training of deep networks. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=BJxsrgStvr`.

[248] Jonathan Young, Marc Modat, Manuel J. Cardoso, Alex Mendelson, Dave Cash, and Sebastien Ourselin. Accurate multimodal probabilistic prediction of conversion to alzheimer's disease in patients with mild cognitive impairment. *NeuroImage: Clinical*, 2:735–745, 2013. doi: 10.1016/j.nicl.2013.05.004. URL `https://doi.org/10.1016/j.nicl.2013.05.004`.

[249] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*, 2017.

[250] Haonan Yu, Sergey Edunov, Yuandong Tian, and Ari S Morcos. Playing the lottery with rewards and multiple languages: lottery tickets in rl and nlp. *arXiv preprint arXiv:1906.02768*, 2019.

[251] Wenchao Yu, Wei Cheng, Charu C Aggarwal, Kai Zhang, Haifeng Chen, and Wei Wang. Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2672–2681, 2018.

[252] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[253] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):1943–1955, 2015.

[254] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *Advances in Neural Information Processing Systems*, 2019.

[255] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. Dynamic network embedding by modeling triadic closure process. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[256] Ligeng Zhu. Thop: Pytorch-opcounter. https://github.com/Lyken17/pytorch-OpCounter, 2019.

[257] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 875–886, 2018.