Artificial Intelligence Review (2019) 52:527–561
https://doi.org/10.1007/s10462-019-09706-7

A review of modularization techniques in artificial neural networks

Mohammed Amer[1] · Tomás Maul[1]

Published online: 4 April 2019

Abstract
Artificial neural networks (ANNs) have achieved significant success in tackling classical and modern machine learning problems. As learning problems grow in scale and complexity, and expand into multi-disciplinary territory, a more modular approach for scaling ANNs will be needed. Modular neural networks (MNNs) are neural networks that embody the concepts and principles of modularity. MNNs adopt a large number of different techniques for achieving modularization. Previous surveys of modularization techniques are relatively scarce in their systematic analysis of MNNs, focusing mostly on empirical comparisons and lacking an extensive taxonomical framework. In this review, we aim to establish a solid taxonomy that captures the essential properties and relationships of the different variants of MNNs. Based on an investigation of the different levels at which modularization techniques act, we attempt to provide a universal and systematic framework for theorists studying MNNs, also trying along the way to emphasise the strengths and weaknesses of different modularization approaches in order to highlight good practices for neural network practitioners.

Keywords Artificial neural network · Modularity · Architecture · Topology · Problem decomposition · Taxonomy

1 Introduction

Modularity is the property of a system whereby it can be broken down into a number of relatively independent, replicable, and composable subsystems (or modules). Although modularity usually adds overhead to system design and formation, it is often the case that a modular system is more desirable than a monolithic system that consists of one tightly coupled structure.

Each subsystem or module can be regarded as targeting an isolated subproblem that can be handled separately from other subproblems. This facilitates collaboration, parallelism

✉ Mohammed Amer
hcxma1@nottingham.edu.my

Tomás Maul
tomas.maul@nottingham.edu.my

1 School of Computer Science, University of Nottingham Malaysia Campus, Semenyih, Malaysia

and integrating different disciplines of expertise into the design process. As each module is concerned with a certain subtask, the modules can be designed to be loosely coupled, which enhances its fault tolerance. Also, a modular design with well defined interfaces makes it easier to scale and add more functionality without disrupting existing functions or the need for redesigning the whole system. Moreover, as modules correspond to different functions, error localization and fixing tend to be easier.

A modular neural network (MNN) is a neural network that embodies the concepts and practices of modularity. Essentially, an MNN can be decomposed into a number of subnetworks or modules. The criteria for this decomposition may differ from one system to another, based on the level at which modularity is applied and the perspective of decomposition.

Since the inception of artificial neural networks and throughout their development, many of their design principles, including modularity, have been adapted from biology. Biological design principles have been shaped and explored by evolution for billions of years and this contributes to their stability and robustness. Evolutionary solutions are often innovative and exhibit unexpected shortcuts or trade-offs that, even if not directly implementable, often provide useful insights. Mapping from biological principles to in-silico realizations is not a linear one-to-one process. However, there are several common steps that can be shared by different such realizations. Ideally, the process of adapting a biological design principle to an artificial neural network implementation starts with identifying the key function(s) underlying the design principle. Usually, there are several complex biological details that are irrelevant to the functional essence of the principle, which can thus be abstracted away. This may be followed by some enhancement of the identified function(s) in the artificial domain. Finally, the abstracted and enhanced principle is mapped to an artificial neural network construct using a flexible platform. Convolutional neural networks (CNNs) (Lecun et al. 1998; Fukushima et al. 1983) are a poignant success story of the adoption of some of the key design principles of the visual cortex. The model of the visual cortex was greatly simplified by CNNs by eliminating complexities like the existence of different cortical areas (e.g. areas V1 and V2) and pathways (e.g. ventral and dorsal streams) and focusing on receptive field, pattern specific regions and a hierarchy of extracted features. These were realized using linear filters, weight sharing between neurons and deep composition of layers. More examples include recurrent neural networks (RNNs), which are inspired by the brain's recurrent circuits (Douglas and Martin 2007), and parallel circuit (PC) neural networks (Phan et al. 2015), which take their inspiration from retinal microcircuits (Gollisch and Meister 2010).

Biological nervous systems, the early inspiration behind neural networks, exhibit highly modular structure at different levels, from synapses (Kastellakis et al. 2015), to cortical columnar structures (Mountcastle 1997), to anatomical (Chen et al. 2008) and functional (Schwarz et al. 2008) areas at the macro level . It has been proposed that natural selection for evolvability would promote modular structure formation (Clune et al. 2013). Modularity is evolvable as it allows for further evolutionary changes without disrupting the existing functionality. It also facilitates exaptation, where existing structures and mechanisms can be reassigned to new tasks (Kashtan and Alon 2005). Recognition of this prevalence of modularity in biological neural systems has left an indelible albeit somewhat irregular mark on the history of artificial neural networks (ANNs), mostly under the guise of biologically plausible models. However, with the divergence between the fields of ANNs and Neuroscience, the ANN approach has tended to become more engineering oriented, getting most of its inspiration and breakthroughs from statistics and optimization.

Many researchers have long been aware of the importance and necessity of modularity. For example, the ANN community has for many years recognized the importance of constraining neural network architectures in order to decrease system entropy, lower the number of free parameters to be optimized by learning, and consequently, have good generalization performance. In Happel and Murre (1994), it is argued that constraining neural architectures through modularity, facilitates learning by excluding undesirable input/output mappings, by using prior knowledge to narrow the learning search space. In Caelli et al. (1999), modularity is considered a crucial design principle if neural networks are to be applied to large scale problems. In Sharkey (1996) and Xu et al. (1992), some of the early techniques of integrating different architectures or modules to build MNNs are discussed. In Caelli et al. (1999), six different MNN models were analytically dissected, in an attempt to provide several modeling practices. On the other hand, in another comparative study (Auda and Kamel 1998), ten different MNN models were empirically compared using two different datasets. In Auda and Kamel (1999), a survey was done about MNNs, where the MNN design process was broken down into three stages, starting by task decomposition, then training and finally, decision making.

Despite this early interest in neural network modularity, previous research (Waibel 1989; Happel and Murre 1994; Fritsch 1996; Ronco and Gawthrop 1995; Auda and Kamel 1998, 1999; Sharkey 1996; Chris Tseng and Almogahed 2009; Kacprzyk and Pedrycz 2015; LeCun et al. 2015) has generally focused on particular MNN models and has lacked systematic principles and a broad general perspective on the topic. Previous research has also been lacking in terms of a systematic analysis of the advantages and disadvantages of different approaches (Chris Tseng and Almogahed 2009), with an increased focus on empirical comparisons of very specific models and applications (Waibel 1989; Auda and Kamel 1998; Fritsch 1996). Even for theoretically focused reviews, the taxonomy is sparse and fails to capture important properties and abstractions (Ronco and Gawthrop 1995; Auda and Kamel 1999; Sharkey 1996; Kacprzyk and Pedrycz 2015). Moreover, the scope of modularity focused on is very narrow, ignoring important forms of modularity and focusing mainly on ensembles and simple combinations of models (Sharkey 1996; Happel and Murre 1994; Fritsch 1996). These limitations need to be addressed if modularity is to be applied more generally. More general insights and a toolbox of modularity-related techniques are needed for consistently implementing successful MNNs. Fortunately, recent MNN techniques have been devised and revisited, specially in the last decade after the revival of the ANN field in the form of deep learning.

In this review, we aim to expand previous reviews by introducing and analysing modularization techniques in the neural networks literature in an attempt to provide best practices to harness the advantages of modular neural networks. We reviewed prominent modular neural networks throughout the literature, inspected the different levels at which modularity is implemented and how this affects neural network behaviour. We then systematically grouped these techniques according to the aspect of neural networks they exploit in order to achieve modularity. Unlike previous reviews, our focus is the general systematic principles that governs applying modularity to artificial neural networks and the advantages and disadvantages of the different techniques. We produced a general taxonomy that captures the major traits of different modular neural networks at different levels and for various modularity forms and a framework that captures the essentials of the process of building a modular neural network.

From our study of modular neural networks in the literature, we classified modularization techniques into four major classes, where each class represents the neural network attribute manipulated by the technique to achieve modularity. We thus categorized MNN operations into the following four classes:

1. Domain: this is the input space or the data an MNN operates on, which in turn defines and constrains the problem we are trying to address.

2. Topology: this corresponds to an MNN's architecture, which reflects the family of models that an MNN expresses.
3. Formation: this is how an MNN is constructed and what process is used to form its components.
4. Integration: this is how the different components of an MNN are composed and glued together to form a full network.

So, modularization techniques operating on the domain tend to act by finding a good partitioning of the input data, to which different modules can be assigned. This is the only modular level that is optional in the sense that you may have an MNN that doesnt have an explicit modularization of the domain, however, any neural network that is modular must use at least one technique from each successive level, which includes selecting a certain modular topology, a formation technique for building the modular architecture, and an integration scheme for combining the different modules. So, as mentioned, topological modularization is the next level at which modularity is achieved, where the technique is essentially a specification of modular topology. Every topological technique is a blueprint for the structure of the MNN, and therefore defines how nodes and modules are connected. Although the topological technique specifies how the MNN as a whole should be at the end, it doesnt specify how this architecture can be built. This is what formational techniques try to address. Formational techniques are the processes by which modular topologies can be constructed. Finally, while formational techniques focus on the building of modularity, integration techniques specify how different modules can be integrated together to achieve the desired system outputs. So, every modular neural network realization can be seen as chain of modularization techniques applied to each level or aspect of the network.

In Sect. 2, we discuss modularity in the context of both ANNs and biological neural circuits in general, along with the different approaches for detecting and quantifying it, its evolutionary context and the practical importance and challenges in applying to engineering problems. In Sect. 3, we discuss the different modularization techniques applied to the different levels of designing an ANN and how different modularization chains can produce a variety of MNNs. In Sect. 4, we analyse different state-of-the-art MNNs, applying our conceptual framework to show its explanatory power and emphasise its practical applicability.

2 Modularity

In the domain of neural networks, modularity is the property of a network that makes it decomposable into multiple subnetworks based on connectivity patterns. It can be argued that the shift of thinking towards functional modularity in the brain and biological neural networks, is one of the greatest leaps in Neuroscience since the neuron doctrine (López-Muñoz et al. 2006). The concept of emphasising the importance of relative connections between neurons and that functionality emerges from intra-modular and inter-modular interactions revolutionized the way we research nervous systems and transformed the idea of a connectome (Bullmore and Bassett 2011; Sporns 2011) into a key area of brain research.

As already mentioned, the brain has been shown to be modular at different spatial scales, from the micro level of synapses to the macro level of brain regions. At the level of synapses, it has been suggested (Kastellakis et al. 2015) that synapses show both anatomical and functional clustering on dendritic branches, and this plays a central role in memory formation. At a larger spatial scale, cortical minicolumns (Buxhoeveden 2002) have been suggested to be the basic building unit of the cortex, largely supported by the claim that they have all of the elements of the cortex represented within them. Lesion studies, brain imaging using fMRI and several other techniques have shown strong evidence of brain modularity, where different areas and regions of the brain are specialized into certain cognitive or physiological functions. More recently, the pioneering work by Sporns, Bullmore and others and the introduction of graph theory into the study of brain networks have shed light on the small world nature of brain connectivity (Bullmore and Sporns 2009; Sporns and Zwi 2004). A small world network is a network characterised mainly by clusters which are groups of neurons having more interconnections within the cluster than would be expected by chance, while there is still sparse connectivity between different such clusters. Moreover, despite large networks and sparse inter-cluster connectivity, there is still a short average path length between neurons. In Sporns and Zwi (2004), it was observed that there is a direct correlation between clustering and path length, and between these two measures and brain area functionality. It was suggested that areas with short average path length and low clustering tend to be integrative multimodal association areas, while those with long average path length and high clustering tend to be specialised unimodal processing areas.

The graph theoretical approach to studying neural networks considers the network as a connected graph, where the neurons are represented by nodes (or vertices) and the synaptic connections between neurons as edges. Although, in practise, neural networks are directed graphs, i.e. edges have directionality from pre to postsynaptic neurons, for simplicity and tractability, most researchers in this area treat neural networks as undirected graphs. Central to quantifying the small world properties of biological neural networks is how to cluster or partition the nodes into modules, where each module has dense connectivity between its nodes and sparse connectivity with nodes in other modules. There is no single most efficient algorithm for solving this problem, and indeed it was proven to be an NP-complete problem (Brandes et al. 2008), however, similar problems have long been studied in Computer science and Sociology (Newman 2004, 2006).

In the field of computer science, graph partitioning is a well studied problem, where given a certain graph and pre-specified number of groups, the problem is to equally partition the vertices into the specified number of groups, whilst minimizing the number of edges between groups. The problem was motivated by other applications before the interest in partitioning neural networks, like partitioning tasks between parallel processors whilst minimizing inter-processor communication. The main approach in computer science is a collection of algorithms known as iterative bisection, such as spectral bisection and Kernighan–Lin algorithm. In iterative bisection, first the graph is partitioned into the best two groups, then subdivisions are iteratively made until the desired number of groups is reached. The problem with these methods is that the number and sizes of groups are not known a priori when partitioning neural networks. Moreover, a lack of good partitioning measures leads these algorithms to deterministically partition the graph into the desired number of groups, even if the partitions dont reflect the real structure of the graph.

On the other hand, sociological approaches have focused more on the problem of community structure detection, which is more suited to neural network research. Community structure detection consists of the analysis of a network in an attempt to detect communities or modules, where the algorithm does not pre-specify the number or size of groups. In other words, it is an exploratory approach, where the algorithm may detect subgraphs or may signal that the graph is not decomposable. The main technique used so far in sociological studies is hierarchical clustering. Based on a metric called similarity measure, hierarchical clustering constructs a tree-like structure of network components called a dendrogram. The horizontal section of this dendrogram at any level gives the network components produced by the algorithm. The algorithm doesnt require pre-specification of the number or sizes of groups, but it doesnt necessarily guarantee the best division.

In more recent approaches (Newman 2004; Tyler et al. 2005; Radicchi et al. 2004), a modularity measure (Eq. 1) was used to either guide the detection process towards the best division or evaluate the quality of the resulting partitioning. The intuitive notion of modularity as defined by Newman (2004, 2016) is that a good network division is one that places most of the network edges within groups, whilst minimizing the number of edges between groups. Network connectivity is assumed to be described by a real symmetric matrix called adjacency matrix $\boldsymbol{A}$, with dimensions $n \times n$, where n is the number of nodes in the network. Each element $\boldsymbol{A}_{ij}$ is 1 if there is an edge between node i and j and 0 otherwise. If we assume dividing the network into q number of groups, where g_i refers to the group to which node i was assigned, then the sum of edges within groups (i.e between nodes of the same group) is $\frac{1}{2}\sum_{ij}\boldsymbol{A}_{ij}\delta_{g_i g_j}$, where $\delta_{g_i g_j}$ is the Kronecker delta. Maximizing this quantity alone is no guide towards a good division, because assigning all the nodes to one big group would maximize this measure whilst completely avoiding any partitions. To remedy this, modularity is taken to be the difference between this quantity (i.e the actual sum of edges within groups) and the expected number of this sum if edges were placed randomly, whilst keeping the same partition. If the probability of node i connecting to node j after randomization is P_{ij}, then this expected sum is $\frac{1}{2}\sum_{ij} P_{ij}\delta_{g_i g_j}$, and the modularity measure is then

$$Q = \frac{1}{2m}\sum_{ij}(\boldsymbol{A}_{ij} - P_{ij})\delta_{g_i g_j} \tag{1}$$

where m is the total number of edges, which is used as a normalization factor. The most used randomization scheme is one that preserves node degree (i.e number of edges attached to each node), and the probability of connecting node i and j under this scheme is $\frac{k_i k_j}{2m}$, where k_i is the degree of node i. Please refer to Fig. 1 for an illustration of different neural topologies with corresponding modularity measures.

On the evolutionary side, multiple hypotheses have been proposed for explaining the origin of modularity in brain organization. The issue is important from the ANN perspective, as it provides inspiration for guiding evolutionary computational algorithms towards generating modular architectures. It was suggested that evolution in an environment with Modularly Varying Goals (MVGs) leads to modular networks (Kashtan and Alon 2005). The MVG environment consists of varying goals with common subgoals. As the modularity of the solution obtained was usually limited, it was argued that the failure might be explained by the fact that the evolutionary algorithm was directed more towards optimal solutions, which was sufficient to solve simple problems, but due to lack of evolvability, failed to scale up to more complex problems. In other studies, the competition between efficient information transfer and wiring cost of the brain have been suggested as sufficient evolutionary pressures for modularity (Clune et al. 2013; Bullmore and Sporns 2009). It was also suggested that selection for minimal connection cost bootstrapped modularity, while MVG helped in maintaining it (Clune et al. 2013).

Using multiple modules in practice is partly motivated by the existence of different subproblems, that may have different characteristics, promoting problem decomposition and functional separation of tasks, that typically contribute towards maintainability and ease of debugging. There are, however, difficulties that surround applying modular neural networks to practical problems. First of all, domain decomposition into meaningful subproblems is usually difficult as the problems tackled by neural networks are usually poorly understood. Moreover, adding modularity to a neural network tends to add a number of new hyperparam-

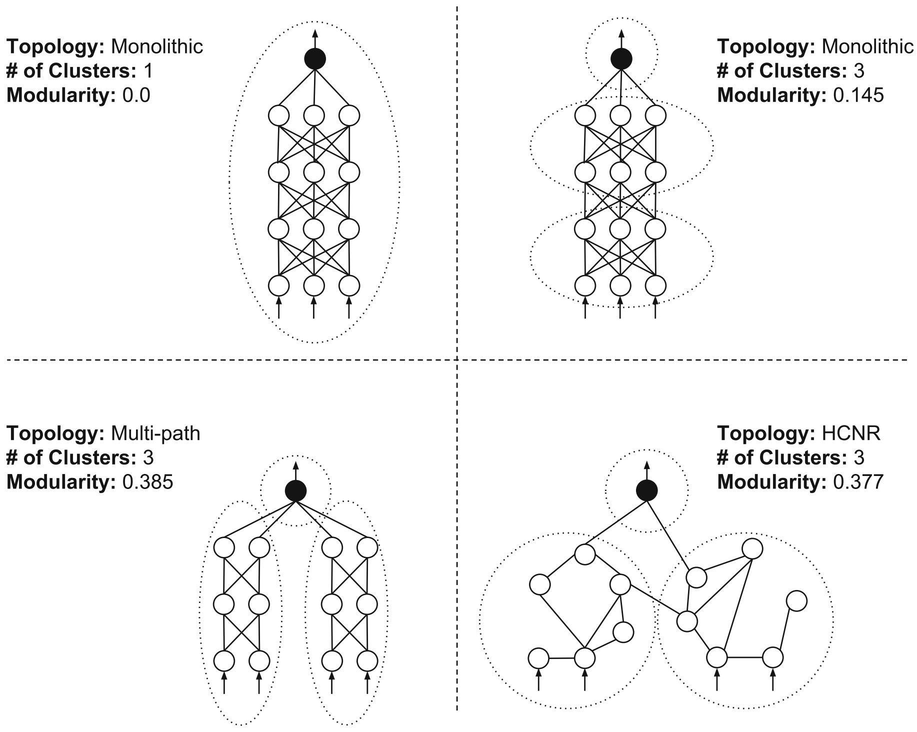

Fig. 1 Calculated modularity measure (Newman 2004, 2016) for different architectures. The partitions, marked as circles, used for calculations may not be optimal

eters that need optimizing, such as the number of modules, the size of each module, and the pattern of connectivity between modules. Another problem that arises with multiple modules is how to integrate the output of different modules and how to resolve any decision conflicts that might arise. We try to address these different problems throughout this study. We discuss the issues surrounding domain decomposition in Sect. 3.1, where we show that problem decomposition can be done implicitly or explicitly, and indicate how the process can be automated. Hyperparameter selection and associated issues are discussed in Sect. 3.3, where the different techniques for MNNs formation are presented. Integrating different modules to solve the task at hand is further investigated in Sect. 3.4.

While the study of modularity has focused mainly on topology, which is indeed the main property of modular structure, we expand our study of modularization techniques to different levels of neural network design that can be exploited to produce modular networks. In the following section, we discuss the different levels of modular neural networks, and how different chains of techniques applied to such levels can produce different MNN variants.

3 Modularization techniques

The modularization of neural networks can be realized with different techniques, acting at different levels of abstraction. A modularization technique is a technique applied to one of these levels in order to introduce modularity to the network topology. We present a taxonomy of such techniques that are categorized based on the abstraction level they exploit to achieve modularity. We analyse each technique, explaining the main rationale behind it, presenting its advantages and disadvantages (Table 1) relative to other techniques and providing prominent use cases from the literature. The main levels at which the modularization techniques act are complementary. Consequently, to produce a modular neural network, a chain of techniques, or chain of modularization, is used. A modularization chain (Fig. 2) consists of a set of techniques (each one corresponding to a different level of the neural network environment) used to produce a modular neural network. So, every modularization chain corresponds to a particular type of MNN.

A modularization chain starts with partitioning the domain, however this is optional as was mentioned earlier. Then, a modular topological structure is selected for the model. After that, formation and integration techniques are selected to build the model and integrate the different modules, respectively. So, for example, if we need to develop an MNN for enhanced MNIST classification, then a modularization chain would look like the following:

1. Domain: we may choose to augment the MNIST dataset by applying a certain image processing function to a copy of each image to extract specific information, and then consider the original and processed images as different subdomains.
2. Topology: here we may select a multi-path topology, where one path of the network has the original image as input and the others have the processed ones.
3. Formation: we may use an evolutionary algorithm to build the multi-path topology, constraining it to have exactly two paths.
4. Integration: here we may integrate the outputs of each path into the final system output, either through the evolutionary process itself, or as a post-formation learning (or fine-tuning) algorithm.

The underlying concept in the example above is to use MNNs to integrate different sources of information (i.e source and processed images) to improve classification performance. A similar concrete MNN application was investigated in Ciregan et al. (2012) and Wang (2015).

3.1 Domain

The domain refers to all of the information that is relevant to the problem and is accessible to the neural network learning model. In other words, it consists of the inputs and outputs that the system relies on for learning how to generalize to unseen inputs. Focusing on the input side, one of the rationales behind domain modularization, is that some functions can be defined piecewise, with each subfunction acting on a different domain subspace. So, instead of learning or applying the neural network model on all of the input space, domain modularization aims to partition this space into a set of subspaces. Then, the modules of an MNN, constructed by applying techniques at different modularization levels, can be readily applied to each subdomain. So, for example, we may choose to partition temporal data according to the time intervals in which they were collected, or partition spatial data according to the places in which they occurred like in Vlahogianni et al. (2007). We refer to this kind of domain partitioning as subspatial domain partitioning, because the individual data items are clustered into multiple subspaces. Curriculum learning (Bengio et al. 2009, 2015) is a particular form of this partitioning, where the neural network is successively trained on a sequence of subspaces with increasing complexity. Another kind is what we call feature or dimensional domain partitioning. In feature domain partitioning, partitioning occurs at the level of a data instance, such that different subsets of features or dimensions or transformations of these get assigned to different partitions. Examples of this approach include the application

Table 1 Advantages and disadvantages of different technique

Technique	Advantages	Disadvantages
A. Domain		
1. Manual	– Prior knowledge integration	– Partitions are hard to define
	– Fine control over partitions	– Relation between decomposition and solution is not straightforward
		– Separation of variation factors is hard
e.g Anand et al. (1995), Oh and Suen (2002), Rudasi and Zahorian (1991), Subirats et al. (2010), Bhende et al. (2008), Mendoza et al. (2009a, b), Ciregan et al. (2012), Wang (2015), Vlahogianni et al. (2007), Aminian and Aminian (2007)		
2. Learned	– Capture useful relations not tractable by human designer	– Computational cost and extra step of learning the decomposing model
e.g Ronen et al. (2002), Fu et al. (2001), Chiang and Fu (1994)		
B. Topology		
1. HCNR	– Sparse connectivity	– Complex structure
	– Short average path	– Hard to analyse and adapt to problems
		– Formation difficulty
e.g Bohland and Minai (2001), Huizinga et al. (2014), Verbancsics and Stanley (2011), Garcia-Pedrajas et al. (2003), Mouret and Doncieux (2009, 2008)		
2. Repeated block		
2.1. Multi-path	– Parallelizable	– Additional hyperparameters
	– Suitable for multi-modal integration	– Currently lacks theoretical justification
e.g Phan et al. (2015, 2017, 2016), Ortín et al. (2005), Wang (2015), Xie et al. (2016), Guan and Li (2002)		
2.2. Modular node	– Computational capability with relatively fewer parameters	– Additional hyperparameters
	– Can be adapted for hardware implementation	
e.g Moon and Kong (2001), Jiang and Kong (2007), Serban et al. (2016), Soutner and Müller (2013), San et al. (2011), Karami et al. (2013), Pan et al. (2016), Srivastava et al. (2013), Lin et al. (2013), Eyben et al. (2013), Yu et al. (2016), Hochreiter and Urgen Schmidhuber (1997), Stollenga et al. (2015), Wang et al. (2011), Kaiser and Hilgetag (2010)		
2.3. Sequential	– Deep composition	– Hard training
		– Excessive depth is arguably unnecessary
e.g Szegedy et al. (2015, 2016), Chollet (2016), Srivastava et al. (2015), He et al. (2016)		
2.4. Recursive	– Readily adaptable to recursive problems	– Excessive depth is arguably unnecessary
	– Deep nesting with short paths	
e.g Franco and Cannas (2001), Larsson et al. (2016)		

Table 1 continued

Technique	Advantages	Disadvantages
3. Multi-Architectural	– Better collective performance – Error tolerance	– Computationally complex
e.g Ciregan et al. (2012), Babaei et al. (2010), Shetty and Laaksonen (2015), Yu et al. (2016), Pan et al. (2016), Kim et al. (2017), Weston et al. (2014)		
C. Formation		
1. Manual	– Prior knowledge integration – Fine control over components	– Hard in practice
e.g de Nardi et al. (2006), Huang (2003)		
2. Evolutionary	– Adaptable way for modularity formation – Suitable for HCNR formation	– Lengthy and computationally complex
e.g Huizinga et al. (2014), Garcia-Pedrajas et al. (2003), Braylan et al. (2015), Miikkulainen et al. (2017), Reisinger et al. (2004), Hüsken et al. (2002), Calabretta et al. (2000), Di Ferdinando et al. (2001)		
3. Learned	– Dynamic formation of modularity – Sample from large set of models	– Computational complexity – In implicit learned variant, networks are densely connected
e.g Srivastava et al. (2014), Huang et al. (2016), Singh et al. (2016), Larsson et al. (2016), Andreas et al. (2016a, b), Hu et al. (2016), Phan et al. (2016, 2017), Blundell et al. (2015)		
D. Integration		
1. Arithmetic-logic	– Prior knowledge integration – Loosely coupled modules	– Difficult in practice
e.g Gradojevic et al. (2009), de Nardi et al. (2006), Wang et al. (2012)		
2. Learned	– Captures complex relations	– Computationally complex – Tightly coupled modules
e.g Zheng et al. (2006), Mendoza et al. (2009b, a), Almasri and Kaluarachchi (2005), Melin et al. (2007), Melin et al. (2011), Hidalgo et al. (2009)		

of different filters to the original images and processing each with different modules like in Ciregan et al. (2012), and the partitioning of an image to enhance object detection accuracy (Zhang et al. 2014).

The domain is, conceptually, the most natural and straightforward level of modularization. This is essentially because the domain defines the problem and its constraints, so, a good modularization of the domain corresponds directly to good problem decomposition. Decomposition of complex problems greatly simplifies reaching solutions, facilitates the design and makes it more parallelizable in both conception and implementation. In de Nardi et al. (2006), the problem of replacing a manually designed helicopter control system by a neural network couldn't be tackled when a single MLP was trained to replace the whole system. However, it was feasible by replacing the system components gradually. Moreover, as it holds all the available information about the problem and its structure, it acts as a very

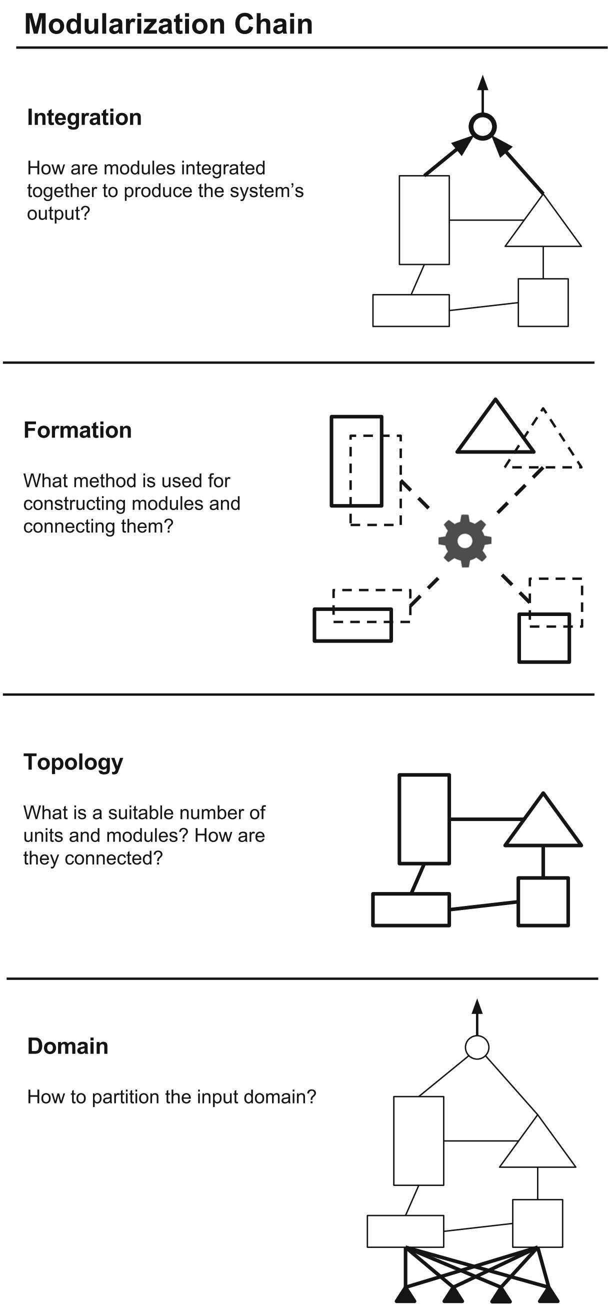
Modularization Chain
Integration
How are modules integrated together to produce the system's output?
Formation
What method is used for constructing modules and connecting them?
Topology
What is a suitable number of units and modules? How are they connected?
Domain
How to partition the input domain?

Fig. 2 Modularization chain acting on the different levels of a neural network

good hook for integrating prior knowledge, through the implementation of modularity, that may be useful in facilitating problem solving. Prior knowledge at the domain level is mainly a basis for problem decomposition, be it an analytical solution, some heuristic or even a learning algorithm. For example, in Babaei et al. (2010), the problem domain of predicting protein secondary structure was decomposed into two main groups of factors, namely: strong correlations between adjacent secondary structure elements and distant interactions between amino acids. Two different recurrent neural networks (RNNs) were used to model each group of factors before integrating both to produce the final prediction. It is also interesting to note that the domain is the only level of modularization that can be absent, at least explicitly, from a modular neural network. In other words, you can have a modular neural network that doesnt involve any explicit modularization at the domain level, but this is not possible for the other levels. This is mainly because domain decomposition is a kind of priming technique for modularity, in other words, it promotes modularity but is not a necessary condition. Note that domain decomposition can still happen implicitly without intentional intervention. As a simple example, it is well established in the machine learning literature that radial basis function (RBF) networks, and also non-linearities in feedforward networks, are able to transform inputs that may be non-linearly separable into linearly separable ones (Haykin 1994; Montufar et al. 2014).

3.1.1 Manual

Manual domain modularization is usually done by partitioning the data into either overlapping or disjoint subspaces, based on some heuristic, expert knowledge or analytical solution. Theses partitions are then translated into a full modular solution via different approaches throughout the modularization process.

The manual partitioning of input space allows for the integration of prior knowledge for problems that are easily decomposable based on some rationale. This knowledge-based integration can be done by defining partitions that correspond to simple subproblems that can be addressed separately. Moreover, it gives fine control over the partitioning process that can be exploited to enhance performance. This is contrary to automatic decomposition, which may be adaptive and efficient, but as the rationale is latent and not directly observed, it is hard to tweak manually for further enhancements. On the other hand, it raises the question of what defines a good partition, a partition which corresponds to a well defined subproblem, that has isolated constraints and can be solved separately. Although the domain is what characterises a problem's solution, usually the relation between decomposing the domain and obtaining a solution is not that straightforward. For example, you may think of decomposing some input image to facilitate face recognition. However, since the process of face recognition is not well understood, it is not clear what decomposition is suitable. Is it segmentation of face parts or maybe some filter transformation? (Chihaoui et al. 2016) Figuring this out analytically is not feasible. The data generating process is often very complex and contains many latent factors of variations, which makes the separation and identification of those factors hard. A good partitioning requires a good prior understanding of the problem and its constraints, which is rarely the case for machine learning tasks, which generally rely on large datasets for the automatic extraction of the underlying causal factors of the data.

One of the simplest subspatial partitioning schemas that arises naturally in classification tasks is class partitioning. Class partitioning is the partitioning of the domain based on the target classes of the problem. This is a straightforward approach which is built on the assumption that different classes define good partitions. There are three main class partitioning schemes, namely OAA, OAO and PAQ (Ou and Murphey 2007). In the One-Against-All (OAA) (Anand et al. 1995; Oh and Suen 2002) scheme, the domain of K classes is partitioned into K subproblems, where each subproblem is concerned with how to differentiate a particular class A from its complement, that is, all of the remaining classes which are not A.

One-Against-One (OAO) (Rudasi and Zahorian 1991) partitions the domain into $\binom{K}{2}$ subproblems, with each subproblem concerned with differentiating one class from only one other class. A compromise between the two previous schemes is P-Against-Q (PAQ) (Subirats et al. 2010), where each subproblem aims to differentiate P number of classes from Q number of classes. OAO is the most divisive of the three, which makes it the most computationally expensive, assuming that each classifier's complexity is the same. Whatever the scheme used, the output of each module trained on a different subproblem can then be combined with an integration technique to embody a particular MNN. More generally, different modularization chains can be applied to the different subproblems, thus resulting in different MNNs.

Class partitioning reduces classification complexity and can be seen as a divide-and-conquer approach. If the partitioning results in several smaller datasets, and assuming that the partitioning accurately reflects the problem's underlying structure, then not only should the learning problem be easier, but the overall representation learned by the MNN should be more faithful to the underlying causes of the data. In Bhende et al. (2008), classification of power quality into 11 classes was done by class partitioning using the OAA technique. The MNNs were applied after feature extraction using the S-transform, and the different modules were integrated using a max activation function to produce the final output.

In Vlahogianni et al. (2007), a subspatial domain partitioning, that is not class based, is used to train different modules on forecasting traffic volume at different road locations. In Aminian and Aminian (2007), an electronic circuit is decomposed into multiple subcircuits to facilitate fault detection by several neural network modules. Feature domain partitioning is another form of manual partitioning, and is seen in Mendoza et al. (2009a, b) where edge detection using a fuzzy inference system is done on target images to obtain edge vectors. Then different neural networks are trained on parts of these vectors, and the network outputs are combined using the Sugeno integral to produce the final classification results. Sometimes, the feature domain partitions are just different transformations of the data, with each transformation revealing different perspectives of the data. This is realised in Ciregan et al. (2012) and Wang (2015) where the input image together with its transformations via different image processing functions are used as inputs into a modular Convolutional Neural Network (CNN), in order to enhance classification performance.

3.1.2 Learned

Learned decomposition is the partitioning of the domain using a learning algorithm. Problem domains are often not easily separable. This is closely related to the problem of representation. If the domain can be manually decomposed into an optimal set of subdomains, that is a set of subspaces that capture all of the constraints of the problem compactly, this will significantly facilitate learning. However, usually the data generating process of the domain involves many interacting factors that are not readily observed. Problems like these typically require learning algorithms to be applied both to the partitioning (explicitly or implicitly) and the overall classification problem.

Learned decomposition facilitates the capturing of useful clustering patterns, especially complex ones that are not tractable by human designers. This intractability may stem from different sources like mathematical complexity or poorly understood problems. For example, the prediction of protein secondary and tertiary structure is often a complex and poorly understood process, that may take tremendous amounts of computational resources to simulate (Freddolino et al. 2008; Allen et al. 2001). However, learning algorithms often add computational cost to the overall process, since they typically involve adding an extra step for optimizing the model responsible for the decomposition.

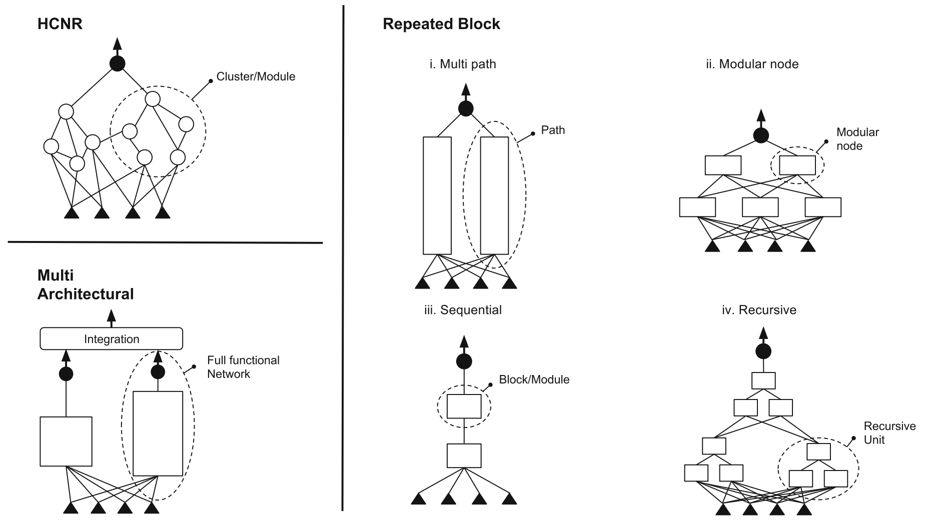

Fig. 3 Different modular neural network topologies

Yuan and Lin (2006) propose modifications to three factor selection methods, namely lasso, LARS and non-negative garrotte, to allow for selecting groups of factors. Effectively, the proposed generalization aims at reaching a good decomposition of the underlying factor space. Because of the clustering nature of learned decomposition, the mainstream approach involves applying unsupervised learning. In Ronen et al. (2002), fuzzy clustering is used to dynamically partition the domain into regions, then a different multilayer perceptron (MLP) is trained on each region, and finally, the MLP outputs are integrated using a Sugeno integral like method. Also, in Fu et al. (2001), a technique called Divide-and-Conquer Learning (DCL) is used to partition the domain whenever learning stalls. DCL acts by dividing training regions into easy and hard sets using Error Correlation Partitioning (ECP) (Chiang and Fu 1994), which is based on optimizing a projection vector that separates data points according to their training error, then different modules are trained on each region and finally integrated using a gated network.

3.2 Topology

The topology of a network refers to how different nodes and modules within the network connect with each other, in order to produce the overall structure of the model. A neural network with a modular topology (Fig. 3) exhibits a structure whereby nodes within a module are densely connected to each other, with sparse connectivity between modules. This is topological modularity, whereas functional modularity emerges when each topological module can be assigned a sub function of the whole task addressed by the neural network model. Topological modularity is a necessary, but not sufficient, condition of functional modularity. Without a learning algorithm that promotes functional modularity, topological modularity is not guaranteed to give rise to functional specialisation.

Neuroscience research sheds light on the modular topology of the nervous system. Neural circuitry in the brain is organized into modules at different levels of granularity, from cortical columns and neuronal nuclei to the whole anatomical areas of the brain's macro structure. It has been suggested in different works that the modularity of the brain arises from selection for evolvability and minimization of connection cost (Bullmore and Sporns 2009; Clune et al. 2013).

Although the early inspiration for neural networks was the brain, artificial neural network research has mostly deviated from biological research. However, there are still occasional insights taken from biology (e.g. deep networks and convolutional structures, to name a few impactful examples) with the usual caveat that the aim is not to closely mimic the brain, but to solve real world problems in effective ways regardless of the source of the core ideas. So, although topological modularity is inspired by the brain's modular structure, it has metamorphosed into different forms that better suit the ANN domain.

The formation and learning of monolithic neural networks are hard problems. This is especially true with very deep neural networks. Deep learning faces several problems like overfitting, vanishing gradients and spatial crosstalk. Good topological modularization acts as a kind of regularization relative to highly connected monolithic networks. For example, Watanabe et al. (2018) provided a method for detecting topological modular structure in trained monolithic neural networks and empirically showed an inverse correlation between the modularity of the detected structure and the network generalization error. Some forms of modular topologies (Larsson et al. 2016; He et al. 2016; Srivastava et al. 2015) provide shortcut paths for gradient flow which help to alleviate vanishing gradients. Moreover, the sparse connectivity of modules reduces spatial crosstalk.

One of the main problems with monolithic networks arises when something wrong occurs. In the vast majority of cases, neural networks are considered black box models. As such, it is usually unrealistically hard to decipher how a neural network makes its predictions. This stems mainly from a neural network's distributed representations, where nodes are tightly coupled, making separation of functions infeasible even in theory. This makes debugging and fixing deviations in behaviour very difficult. Topological modularity, especially if accompanied by functional modularity, can be exploited to localize functional errors so that more investigations may reveal possible solutions. In the case of full functional modularity, there are still distributed representations associated with different modules, however, since modules themselves are loosely coupled, this separation of concerns makes localizing deviation in some sense realistic.

3.2.1 Highly-clustered non-regular (HCNR)

HCNR topology is a modular topology with non-regular and dense within-module connections, and sparse connectivity between different modules. Non-regularity here roughly means that the overall topology cant be described by a template with repeating structures. This makes the topology generally hard to compress. Elements from graph theory can be used to formalize this notion using measures like characteristic path length and clustering coefficient (Watts 1999). Aside from high clustering, HCNR doesnt have to exhibit properties such as the short average path length of small world (SW) networks. Thus, the broader category of HCNR includes small world topologies as special cases.

Biology has shown significant interest in small world networks as increasing evidence suggests that many biological systems including genetic pathways, cellular signalling and brain wiring, exhibit small world topology. The evolutionary origins of the brain's modular structure is still controversial, but some hypotheses have been suggested. In Kashtan and Alon (2005), it was suggested that evolution under a Modularly Varying Goals (MVG) environment yields modular structure. An MVG environment changes modularly, in the sense that the environmental goal varies through time, but each goal is comprised of the same common set of subgoals. A biological example is chemotaxis towards nutrients. The process of chemotaxis involves the same set of intermediate goals, like sensing, computing motion direction and moving, that are independent of the target nutrient. In another hypothesis (Clune et al. 2013) it was suggested that modularity is bootstrapped by a pressure for minimizing costs pertaining to neuronal connections, and is then maintained by selection for evolvability. Random networks tend to have high connection cost due to dense connectivity, which is not the case in the brain's neural circuitry, which exhibits economical functional networks (Achard and Bullmore 2007). Having a modular structure promotes evolvability, as accumulative evolutionary changes tend to be local in effect without disrupting other functions.

In the context of artificial neural networks, the sparse connectivity of HCNR and average short path of its special case SW, reduce computational complexity compared to monolithic networks, whilst maintaining information transfer efficiency. However, due to their structural complexity, analysing and adapting these types of networks to real world problems is hard and raises several technical difficulties. How many nodes should be in each module? Should module node counts vary? How much connectivity is allowed within a module? And what connection sparsity between modules is sparse enough? Also, formation of this network type is done either by modifying a regular lattice (Bohland and Minai 2001), which is hard to adapt to all machine learning tasks, or via evolutionary algorithms (Huizinga et al. 2014; Verbancsics and Stanley 2011; Garcia-Pedrajas et al. 2003; Mouret and Doncieux 2009, 2008), which are lengthy and computationally expensive. The adoption of these two approaches, especially evolutionary algorithms, is a direct consequence of the previously mentioned difficulties and lack of good general engineering practices for HCNR.

The work done in Bohland and Minai (2001) shows that an SW topology can approach the performance of random networks for associative memory tasks, with less connectivity than random networks, implying that associative memories can benefit from modularity. This network was constructed by rewiring a regular lattice randomly. The work by Bohland and Minai (2001) shows that performance is not only about the quantitative nature of connectivity (i.e number of connections), but also about its qualitative nature (i.e how these connections are placed).

Evolutionary approaches for HCNR are either based on direct connection cost regularization, or the coevolution of modules. Both approaches tend to be biologically inspired, with connection cost regularization corresponding to the pressure of minimizing brain wiring, and coevolution inspired by species coevolution, where in this case each module is considered a different species. The evolutionary approach in Huizinga et al. (2014) and Verbancsics and Stanley (2011) made use of the connection regularization studied in biological neural networks (Clune et al. 2013; Bullmore and Sporns 2009) to promote HCNR modularity in the resulting model, which led to better performance on modular regular problems such as the retina problem. Another evolutionary approach relies on the cooperative coevolution model where modules are dependently co-evolved and their fitness is evaluated based on each module's performance and how well each module cooperates with other modules. COVNET (Garcia-Pedrajas et al. 2003), for example, achieved better results in some classification problems, like the Prima Indian and Cleveland Clinic Foundation Heart Disease datasets. COVNET also showed robustness to damage in some network parts.

Currently, there is an increasing interest in learning and manipulating structured representations using Graph Neural Networks (GNNs) (Battaglia et al. 2018). For example, Santoro et al. (2017) introduced a module to enhance solving relational reasoning problems. Message-Passing Neural Networks (MPNNs) (Gilmer et al. 2017) are a type of GNN that were applied to the prediction of properties of molecular structures. Another area of application was recovering textual representations from Abstract Meaning Representations (AMR)

(Song et al. 2018). One application which is very relevant to architectures and topologies is Graph HyperNetworks (Zhang et al. 2018). This is a GNN for predicting a good set of weights for a neural network by analyzing its graph. We believe that GNNs can facilitate the analysis, manipulation and building of complex graphs, like HCNRs, and help understand their properties.

3.2.2 Repeated block

This topology of modular neural networks is essentially a structure of repeated units or building blocks connected in a certain configuration. The building blocks dont have to be exact clones of each other, but they are assumed to share a general blueprint. The idea of global wiring schema in neural networks has its roots in biological studies and it is the underlying principle of the famous neuroevolution algorithm, HyperNEAT (Stanley et al. 2009). In Angelucci et al. (1997) it was shown that retinal projections in ferrets, normally relayed to the Lateral Geniculate Nucleus (LGN), when rewired to the Medial Geniculate Nucleus (MGN), normally a thalamic relay in the auditory pathway, led the MGN to develop eye-specific regions. Also, mammalian cortex is considered to be composed of repeating columnar structure (Lodato and Arlotta 2015). These and other lines of evidence support the notion of a global mechanism of wiring and learning in the brain.

In the artificial realm, repeated block structure allows for easier analysis and extensibility of neural networks. On the theoretical level and due to the high regularity of these topologies, a very large structure can be described by a few equations. For example, a recursive structure like FractalNet (Larsson et al. 2016), can be described by a simple expansion rule. Also, due to regularity, scaling the capacity of these topologies tends to be very natural. We provide a taxonomy of repeated block topologies based on how the repeated units are wired together.
Multi-Path Multipath topology refers to neural networks with multiple semi-independent subnetworks connecting network inputs to outputs. In Phan et al. (2015, 2016, 2017) this topology is named Parallel Circuits (PCs), which are inspired by the microcircuits in the retina (Gollisch and Meister 2010). The retina is believed to be of significant computational importance to the visual pathway, not just a simple informational relay. In other words, it has been shown that the retina does perform complex computational tasks, such as motion analysis and contrast modulation, and delivers the results explicitly to downstream areas. Moreover, these microcircuits have been shown to exhibit some sort of multipath parallelism, embodied by semi-independent pathways involving different combinations of photoreceptor, horizontal, bipolar, amacrine and retinal ganglion cells.

The separation of multiple paths allows for overall network parallelization, contrary to network expansion in terms of depth, where deeper layers depend on shallower ones, which makes parallelization problematic. Also, as in Ortín et al. (2005) and Wang (2015), each path can be assigned to a different input modality which allows for modal integration. This resembles brain organization where different cortical areas process different modalities, and then different modalities get integrated by association areas. However, the introduction of multiple paths adds uncertainty in the form of new hyperparameters (e.g. numbers and widths of paths), which if to be determined empirically, often requires a phase of pre-optimization. Moreover, aside from obvious links to ensemble theory, as of yet there is no detailed theoretical justification for multiple paths. Why do empirical experiments show improved generalization performance (Phan et al. 2016) of multipath over monolithic topologies? Does width (in terms of number of paths) promote problem decomposition just like depth promotes concept composition? To date there are no mature gradient-based learning algorithms to fully exploit the parallel circuit architecture, and which are likely to explicitly promote automatic task decomposition across paths.

In Phan et al. (2015, 2016, 2017), a multipath approach with shared inputs and outputs is shown to often exhibit better generalization than monolithic neural networks. Crucial to this improvement, was the development of a special dropout approach called DropCircuit, where whole circuits are probabilistically dropped during training. In another approach (Guan and Li 2002), called output parallelism, the inputs are shared between paths, while each path has a separate output layer. This technique can be applied when the output is easily decomposable. A very related approach can be found in Goltsev and Gritsenko (2015), where a central common layer is connected to multiple paths, each used for a different output class. On the other hand, the work in Wang (2015) enhances CNNs by allowing for two paths, one with the source image as input, and the other with a bilateral filtered version of it. In bilateral filtering, each pixel value is replaced by the average of its neighbours, taking into account the similarity between pixels, so that high frequency components can be suppressed while edges are preserved. The integration of images preprocessed in different ways facilitates the capturing of more useful features. This is motivated by the observation that convolution and pooling operations extract high frequency components, which causes simple shapes and less textured objects to gradually disappear. Bilateral filtering of one of the input images tends to suppress high frequency components, which allows the network to capture both simple and complex objects. The multi-path concept can be integrated with other topologies, as in Xie et al. (2016) where ResNetXt enhances ResNet's sequential topology by introducing modules that have multi-path structure.

Modular node A modular node topology can be viewed as a normal monolithic feedforward neural network, where each node is replaced by a module consisting of multiple neurons. This expansion is computationally justified by replacing a single activation function depending on one weight vector, by a collection of functions or a function depending on multiple weight vectors. This has the effect of increasing the computational capability of the network, while maintaining a relatively small number of model parameters. Moreover, the regularity and sparsity of such a structure, combined sometimes with restricting weights to integer values, can be suitable for hardware realizations (Moon and Kong 2001). On the other hand, this requires additional engineering decisions, like choosing the number of module neurons, how they are interconnected and what activation functions to use.

A special case of this topology is hierarchical modular topology, which consists of modules at different topological scales, where a higher level module is composed of submodules, each of which is composed of submodules, and so on. Hierarchical modularity is known to exist in brain networks (Wang et al. 2011; Kaiser and Hilgetag 2010), other biological networks and Very Large-Scale Integration (VLSI) electronic chips (Meunier et al. 2010). It has been argued that this form of modularity allows for embedding a complex topology in a low dimensional physical space.

In Moon and Kong (2001), Jiang and Kong (2007) and San et al. (2011) modular node topology is realized by replacing the nodes of a feedforward network by a two dimensional mesh of modules, with modular units each consisting of four neurons. The four neurons can be connected in four different configurations. This network, called Block-Based Neural Network (BBNN), was shown to be applicable to multiple tasks including pattern classification and robotic control, even when its weights were restricted to integer values. Another modular network called Modular Cellular Neural Network (MCNN) (Karami et al. 2013) exhibits similar array like arrangement, where nine modules are arranged in a grid. A module in MCNN is composed of another grid of dynamic cells, where each cell is described by a differential equation. MCNNs were applied to texture segmentation successfully and benchmarked to other algorithms on the problem of edge detection. Another realization of this topology can be found in the Local Winner-Take-All (LWTA) network (Srivastava et al. 2013), where each node of a feedforward neural network is replaced by a block, each consisting of multiple non interconnected neurons. The network operates by allowing only the neuron with the highest activation in a block to fire, while suppressing other neurons. The block output is

$$y_i = g(h_i^1, h_i^2, \dots, h_i^n) \tag{2}$$

where $g(.)$ is the local interaction function and h_i^j is the activation of the jth neuron in block i. This is mainly inspired by the study of local competition in biological neural circuits. Network In a Network or (NIN) models (Lin et al. 2013) are the modular node equivalents of CNNs, where each feature map is replaced by a micro MLP network, to allow for high-capacity non-linear feature mapping. The output of a single micro MLP is

$$\boldsymbol{f_{i,j}^l} = max(0, \boldsymbol{W_l} \boldsymbol{f_{i,j}^{l-1}} + \boldsymbol{b_l}) \tag{3}$$

where (i, j) are indices of the central pixel location, l is the index of the MLP layer and $\boldsymbol{W}$ and $\boldsymbol{b}$ are the weights and bias, respectively. It is also interesting that the long short-term memory (LSTM) architecture (Hochreiter and Urgen Schmidhuber 1997), the famous recurrent neural network (RNN), has a modular node structure, in which each node is an LSTM block. LSTM has shown state-of-the-art results in sequence modeling and different real-world problems (Stollenga et al. 2015; Eyben et al. 2013; Soutner and Müller 2013). Moreover, Hierarchical Recurrent Neural Networks (HRNN), which are typically realised using LSTM or its simplification, the Gated Recurrent Unit (GRU), implements hierarchical modular topology, where the first hidden layer is applied to input sequentially and the layer output is generated every n number of inputs, which is then propagated as input to the next layer and so on. Hence, the main difference between HRNNs and classic RNNs is that, for the former, hidden layer outputs are generated at evenly spaced time intervals larger than one. HRNNs have been used for captioning videos with a single sentence (Pan et al. 2016) and with a multi-sentence paragraph (Yu et al. 2016), and for building end-to-end dialogue systems (Serban et al. 2016).

CapsNet was introduced in Sabour et al. (2017), which is mainly a vision-centric neural network that attempts to overcome the limitations of CNNs. The main rationale behind CapsNet is representing objects using a vector of instantiation parameters that ensures equivariance with different object poses. CapsNet can be thought of as an ordinary CNN, in which each node is replaced by a vector-output module. The output of such a modular node in CapsNet is calculated as

$$\boldsymbol{v_j} = \frac{||\boldsymbol{s_j}||^2}{1 + ||\boldsymbol{s_j}||^2} \frac{\boldsymbol{s_j}}{|}|\boldsymbol{s_j}|| \tag{4}$$

where $\boldsymbol{s_j}$ is the node input and is calculated as

$$\boldsymbol{s_j} = \sum_i c_{ij} \hat{\boldsymbol{u}}_{j|i}, \ \hat{\boldsymbol{u}}_{j|i} = \boldsymbol{W_{ij}} \boldsymbol{u_i} \tag{5}$$

where $\boldsymbol{u_i}$ is the output of a unit i from the previous layer, $\boldsymbol{W_{ij}}$ is a transformation matrix and c_{ij} is a coupling coefficient. Coupling coefficients are calculated through a routing softmax as

$$c_{ij} = \frac{e^{b_{ij}}}{\sum_k e^{b_{ik}}} \tag{6}$$

where b_{ij} is a log prior probability which is initialized to zero and updated following the rule

$$b_{ij} \leftarrow b_{ij} + \hat{\boldsymbol{u}}_{\boldsymbol{j|i}}.\boldsymbol{v_j} \tag{7}$$

This is called routing-by-agreement and acts to increase contributions from lower layer capsules that make good predictions regarding the state of a higher level capsule.
Sequential Sequential topology consists of several similar units connected in series. The idea of composition of units has its roots in deep learning. Deep networks arise when multiple layers are connected in series. This allows for deep composition of concepts, where higher level representations are composed from lower level ones. The difference here is that the composed units consist of whole modules. But with added depth, convergence and generalization can become increasingly difficult, and one must therefore resort to tricks like dropout and batch normalization to make learning feasible. Moreover, there has been recent criticism of very deep networks based on the question of whether this extreme depth is really necessary (Ba and Caruana 2014; Veit et al. 2016), specially given that the brain can do more elaborate tasks with far fewer layers.

Inception networks (Szegedy et al. 2015, 2016) and Xception networks (Chollet 2016) (built from an extreme version of an inception module), are essentially a sequential composition of multi-path convolutional modules. Highway networks were introduced in Srivastava et al. (2015) and can be seen as a sequentially connected block of modules, where each module output consists of the superposition of the layer output and layer input, weighted by two learning functions called the transform gate and carry gate respectively. The output of a single layer in a highway network can be modelled as

$$y = H(x, \boldsymbol{W_H}).T(x, \boldsymbol{W_T}) + x.C(x, \boldsymbol{W_C}) \tag{8}$$

where H is the layer activation function, T is the transform gate and C is the carry gate. These two gates learn to adaptively mix the input and output of each module, acting as a regulator of information flow. A similar idea can be found in He et al. (2016) where a special case of highway networks called residual networks consists of the same structural unit but with both gates set to the identity function. This makes the residual layer output

$$y = H(x, \boldsymbol{W_H}) + x \tag{9}$$

This is motivated by simplifying the learning problem and enforcing residual function learning. Interestingly, the LSTM network, whose modular node topology was discussed above, exhibits a temporal form of sequential topology, where each LSTM block feeds its output to itself through time. So, expanding the LSTM block in time results in a temporal sequential topology, where the output of the LSTM block from the previous time step is considered as an input to the LSTM block in the current time step.
Recursive Networks with recursive topology exhibit nested levels of units, where each unit is defined by earlier units in the recursion. Usually, all of the units are defined by the same template of components wiring. Recursion has a long history and is considered to be a key concept in computer science. Although recursive problems can be solved without explicit recursion, the recursive solution is a more natural one. In theory an infinite structure can be defined in one analytical equation or using a simple expansion rule. Due to their recursive structure, networks with recursive topology are readily adaptable to recursive problems (Franco and Cannas 2001). Recursion also allows for very deep nesting while still permitting short paths, sometimes called information highways (Larsson et al. 2016) that facilitate gradient back-propagation and learning. However, as mentioned earlier, excessive depth is criticised by some researchers and its necessity is becoming increasingly debatable.

FractalNet introduced in Larsson et al. (2016) exploits recursive topology to allow for very deeply nested structure that is relatively easy to learn despite its significant depth. It is inspired by the mathematically beautiful self-similar fractal, where going shallower or deeper in structure yields the same topology schema. It is defined as

$$f_{C+1}(x) = [(f_C \circ f_C)(x)] \oplus [conv(x)] \tag{10}$$

where C is the fractal index, $\circ$ means composition and $\oplus$ is a join operation. It is supposed that the effectively shorter paths for gradient propagation facilitate learning and protect against vanishing gradients. In Franco and Cannas (2001) the parity problem was decomposed recursively and a recursive modular structure was adapted for its solution. Also in this work on the parity problem, it was shown that generalization was systematically improved by degree of modularity, however, it was not obvious if that was a general conclusion applying to all problems.

3.2.3 Multi-architectural

A multi-architectural topology consists of a combination of full network architectures, integrated together via a high-level and usually simple algorithm. Frequently, it is characterized by each component network having its separate output. The different architectures used may be similar (i.e homogeneous) or different (i.e heterogeneous). Architectural differences include, but are not limited to, differences in wiring scheme and activation functions. As different network architectures have different strengths and weaknesses (and make different errors), the integration is usually trying to exploit this diversity in order to achieve a more robust collective performance. Even when networks are similar, diversity can still be achieved since random initialization and stochastic learning makes each network converge differently. However, this usually entails training multiple architectures, which is time consuming and computationally expensive.

One of the early attempts for combining multiple architectures was the mixture of experts systems (Jacobs et al. 1991; Azam 2000), where a gating network chooses which one of multiple networks should respond to a given input. In another approach (Ciregan et al. 2012), a homogeneous model is used where similar CNNs are trained on different types of preprocessing of the same image and their outputs are integrated by averaging. In Yu et al. (2018), a referring expression is decomposed into three components, subject, location and relationship, where each component is processed using a separate visual attention module, which is essentially a CNN, and then the outputs of the different modules are combined. In Babaei et al. (2010), a heterogeneous model consisting of two different RNNs, each modeling different protein structural information, is applied to predicting protein secondary structure. In Zhang et al. (2016), a modular deep Q network is proposed to facilitate transferring of a learned robotic control network from simulation to real environment. By modularising the network into three components, namely perception, bottleneck and control modules, the perception training can be done independently from the control task, while maintaining consistency through the bottleneck module acting as an interface between the two other modules. In Shetty and Laaksonen (2015), Yu et al. (2016) and Pan et al. (2016) heterogeneous models of CNNs and RNNs are used for video captioning, where one or more CNNs are used for extracting features, which are used as inputs to RNNs for generating video captions. Generative Adversarial Networks (GANs) and their variants (Goodfellow et al. 2014; Kim et al. 2017) are also multi-architectural in nature. Another interesting example is the memory network (Weston et al. 2014), where multiple networks are composed end-to-end around a memory module to allow for the utilisation of past experiences. Essentially, a memory network is composed of a memory and four components, namely, I, G, O and R. I is the input network that translates the raw input into an internal representation $I(x)$. G stands for generalization and it is responsible for updating memory based on the new input

$$m_i = G(m_i, I(x), m) \tag{11}$$

where i is the index of the memory cell. After the memory is updated, another module, O, computes the output features based on the new input and the memory

$$o = O(I(x), m) \tag{12}$$

and finally, the R module converts the output features into the desired format

$$r = R(o) \tag{13}$$

3.3 Formation

Formation refers to the technique used to construct the topology of the neural network. Manual formation involves expert design and trial and error. In manual formation, the human designer is guided by analytical knowledge, several heuristics and even crude intuition. Because of the difficulty and unreliability of manual formation, and a general lack of understanding of the relation between problems and the models they require, automatic techniques have been devised. Arguably the most popular automatic techniques are evolutionary algorithms, where the structure of the network is evolved over multiple generations, based on a fitness function that evaluates which individuals are more adapted. Another set of automatic formation algorithms constitute the learned formation category, where a learning algorithm is used not only for parameter (e.g. connection weight) optimization, but also for structure selection. Learned formation can be categorized into constructive and destructive algorithms (Garcia-Pedrajas et al. 2003). In constructive learned formation, the algorithm starts with a small model, learns until the performance stalls, adds components to expand capacity and iterates again. Destructive learned formation algorithms start with a big model that overfits, then iteratively remove nodes until the model generalizes well.

In order to form a modular neural network, one of these construction approaches needs to be modified in order to take modularization into account. With manual formation, it is in principle straightforward to modularize, where instead of designing a monolithic network, different modules are designed and combined to build an MNN. On the other hand, while standard evolutionary algorithms can produce modular structure, they are usually modified using techniques like cooperative coevolution, given that the latter are generally seen to be more effective for evolving modular structure. In the case of learned formation, learning algorithms usually take modularity explicitly into account. So, the machine learning task becomes that of learning both modular structure and the parameters (e.g. weights) of that structure. A variant of learned formation, which we call implicit learned formation, is a learning algorithm that is implicitly sampling or averaging from a set of modules, so that the overall effective structure of the network can be seen as a modular one.

3.3.1 Manual

In manual formation, modular networks are built by manual design and composition of different modules. This type of formation provides useful opportunities for integrating good engineering principles and prior knowledge of the target problem into the modular neural network. For example, in Babaei et al. (2010), the system for predicting protein secondary structure is formed from two RNN modules that model two different aspects of the process, namely, short and long range interactions. Fine control over what to include or exclude from the model can lead to a robust combination of well performing components. However, regardless of how this sounds theoretically plausible, limited understanding of the underlying structures of most real-world problems and limited, to date, research into good neural modularization practices, make this hard in practice.

In de Nardi et al. (2006), different modules are manually composed together to implement a helicopter control system, based on the practices of human designed ProportionalIntegralDerivative (PID) controllers. The PID components are replaced progressively by their neural network counterparts, until the whole control is done by the MNN. More formally in Huang (2003), analytical introduction of modular layers into feedforward neural networks allows for reducing the number of nodes required to learn a target task. This is a case that shows how good engineering could be integrated, through formal analysis, into the formation of modular neural networks.

3.3.2 Evolutionary

Evolutionary algorithms represent the current state of the art in formation methods for modular neural networks. This is clearly biologically inspired by the neuroevolution of the brain, which is shown to be highly modular both in topology and functionality (Aguirre et al. 2002; Bullmore and Sporns 2009). Aside from biological inspiration, evolving modular structure has gained momentum as an effective approach to modularity formation, partly because of a lack of fundamental learning principles supporting artificial neural modularity. Adapting evolution to the problem of modularity formation, through connection cost regularization (Huizinga et al. 2014) or cooperative coevolution (Garcia-Pedrajas et al. 2003), partly delegates the problem of choosing modularity-related hyperparameters, such as the number and structure of modules, or connection schema, to a fitness function. Furthermore, evolutionary algorithms are the only fitness-based approach to producing HCNR topology, whereas other methods rely on random modifications to regular networks. On the down side, as already mentioned, evolutionary algorithms tend to be computationally expensive.

In Garcia-Pedrajas et al. (2003), COVNET was introduced, which is a modular network formed using a cooperative coevolutionary algorithm. Every module is called a nodule, and is defined as a set of neurons that are allowed to connect to each other and to input/output nodes, but are not allowed to connect to neurons in other nodules. Every nodule is selected from a genetically separated population and different nodules are combined together to form individuals of the network population. To achieve cooperative coevolution, it is not sufficient to assign fitness values to networks, but it is also necessary to assign fitness values to nodules. The fitness of a network is straightforward, where obviously it corresponds to how well the network performs on its target task. The fitness assignment of nodules must enforce: (1) competition, so that different subpopulations dont converge to the exact same function, (2) cooperation, so different subpopulations develop complementary features and (3) meaningful contribution of a nodule to network performance, such that poorly contributing nodules are penalized. In COVNET, a combination of different measures is used to satisfy these criteria. Substitution is used to promote competition, where the best k networks are selected and a nodule a is replaced by a nodule b from the same subpopulation; then the networks fitnesses are recalculated, and nodule a is assigned fitness proportional to the average difference between the network fitnesses with nodule a and the network fitnesses with the substitution nodule *b*. Difference is used to promote cooperation between nodules by promoting competition between nodule subpopulations, so that they don't develop the same behaviour. Difference is done by eliminating a nodule *a* from all the networks where it is present, then recalculating network fitnesses; then the nodule is assigned fitness proportional to the average difference between fitnesses of the networks with the nodule and the networks without it. Finally, best *k* is used to assess the meaningful contribution of a nodule, where nodule fitness is proportional to the mean of the fitnesses of the best *k* networks. This has the effect of rewarding nodules in the best performing networks, whilst not penalizing a good nodule in a poor performing network.

Promoting modularity through connection cost minimization (Huizinga et al. 2014) is biologically inspired as evidence suggests that the evolution of the brain, guided by the minimization of wiring length, besides improving information transfer efficiency, produces modular structure (Clune et al. 2013; Bullmore and Sporns 2009). In Hüsken et al. (2002), modularity emerges through evolution by selection pressure for both fast and accurate learning. In Di Ferdinando et al. (2001), a modular multi-path neural network was evolved for solving the what and where task of identifying and localizing objects using a neural network. In another type of approach, modules are used as substrates for evolutionary algorithms. For example, in Braylan et al. (2015) pre-learned networks (modules) were combined using evolutionary algorithms, in an attempt to implement knowledge transfer. In Calabretta et al. (2000), evolution was implemented using a technique called duplication-based modular architecture, where the architecture can grow in the number of modules by mutating a set of special duplicating genes. In Miikkulainen et al. (2017) a population of blueprints, each represented by a graph of module pointers, was evolved using CoDeepNEAT, alongside another population of modules, evolved using DeepNEAT, an algorithm based on NEAT (Stanley and Miikkulainen 2002), to develop deep modular neural networks. CoDeepNEAT seems to be a generalization of a previous algorithm called ModularNEAT (Reisinger et al. 2004), where modules are evolved using classic NEAT and blueprints are shallow specifications of how to bind modules to the final network input and output.

In Fernando et al. (2017), an interesting approach to evolutionary formation is introduced, where only some pathways in a large neural network composed of different modules are trained at a given time. The aim is to achieve multi-task learning. The pathways are selected through a genetic algorithm that uses a binary tournament to choose two pathways through the network. These pathways are then trained for a number of epochs to evaluate their fitness. The winner genome overwrites the other one and gets mutated before repeating the tournament. At the end of the training for some task, the fittest pathway is fixed and the process is repeated for the next task.

3.3.3 Learned

Learned formation is the usage of learning algorithms to induce modular structure in neural networks. Learned formation attempts to integrate structural learning into the learning phase, such that the learning algorithm affects network topology as well as parameters. We identified two variants of learned formation in the literature. Explicit learned formation uses machine learning algorithms to promote modularity, predict the structure of modular neural networks and specify how modules should be wired together. On the other hand, implicit learned formation corresponds to learning algorithms that implicitly sample from multiple modules during training, although during the prediction phase, the network is explicitly monolithic whilst effectively simulating a modular network. Learned formation, just like evolutionary formation, allows for dynamic formation of modules. Moreover, as mentioned above, it can effectively sample from a large set of models, which is why it is often referred to as effectively implementing ensemble averaging (Srivastava et al. 2014; Huang et al. 2016; Singh et al. 2016; Larsson et al. 2016). The main disadvantage for dynamic algorithms like these is added computational overhead. Also, for implicit learned formation, the network is still densely connected and therefore computationally expensive, and modules are generally sampled randomly without any preference for better modules.

In Andreas et al. (2016a, b), Hu et al. (2016), which exemplifies recent work on explicit learned formation, the problem of relating natural language to images was addressed. A set of modular blocks, each specialised in a certain function (e.g. attention and classification), were used as building units of a modular neural network, where different dynamic techniques were applied to assemble units together into an MNN that was capable of answering complex questions about images and comprehending referential expressions. For example, in Andreas et al. (2016b), two distributions were defined: (1) layout distributions, defined over possible layouts/structures of the network and, (2) execution model distributions, defined over selected model outputs. The overall training was done end-to-end with reinforcement learning. Another approach (Ferreira et al. 2018) proposes using false nearest neighbours (FNN), an adaptive learning algorithm usually used to determine the dimensionality of embedding, to help determine the kernel size and number of units for CNNs.

One of the most well known implicit learned formation techniques is dropout (Srivastava et al. 2014). Dropout acts by dropping random subsets of nodes during learning, as a form of regularization that prevents interdependency between nodes. Dropout is effectively sampling from a large space of available topologies during learning, because each learning iteration acts on a randomly carved sparse topology. In the prediction phase, networks are effectively averaging those random topologies to produce the output. Stochastic depth (Huang et al. 2016) is another dropping technique used in training residual networks, which acts by dropping the contribution of whole layers. Swapout (Singh et al. 2016) generalizes dropout and stochastic depth, such that it is effectively sampling from a larger topological space, where a combination of dropping single units and whole layers is possible. DropCircuit (Phan et al. 2016, 2017) is another related technique, which is an adaptation of dropout to a particular type of multipath neural network called Parallel Circuits. In this technique, whole paths are randomly dropped, such that learning iterations are acting on random modular topologies. This effectively corresponds to sampling from a topological space of modular neural networks. Blundell et al. (2015) introduced Bayes by Backprop, a learning algorithm that approximates Bayesian inference which is, as applied to a neural network, the sum of the predictions made by different weight configurations, weighted by their posterior probabilities. This is essentially an ensemble of an infinite number of neural networks. As this form of expectation is intractable, it is approximated using variational learning, using different tricks such as Monte Carlo approximation and a scale mixture prior.

3.4 Integration

Integration is how different module outputs are combined to produce the final output of the MNN. Integration may be cooperative or competitive. In cooperative integration, all the MNN modules, contribute to the integrated output. On the other hand , competitive integration selects only one module to produce the final output. The perspective of integration is different from that of formation, where the latter is concerned with the processes that gives rise to modular structure, and the former is concerned with the structures and/or algorithms that use different modules in order to produce model outputs. Integration is a biologically inspired theme of brain structure, where hierarchical modular structures work together to solve a continually changing set of complex and interacting environmental goals.

3.4.1 Arithmetic-logic

Arithmetic-Logic (AL) integration corresponds to a set of techniques that combine different modules through a well-defined algorithmic procedure, combining mathematical operators and logic. For problems that can be described using a sequence of algorithmic steps, this is the simplest and most straightforward approach, and is the most natural hook for integrating prior knowledge. It is worth mentioning that while the relation between steps needs to be algorithmically defined, the computation of the steps themselves is not necessarily well-defined. For example, a car control system may want to steer away from an obstacle once it has identified one. The relation between identification and steering away is AL-defined, while the identification of obstacles is not generally algorithmically defined. Moreover, AL integration allows for module decoupling, where each module has its well-defined interpretable output, which further makes debugging easy. However, in machine learning tasks, due to our limited understanding of problem domains and corresponding data generating processes, it is rarely the case that problems can easily be decomposed into AL steps.

In Gradojevic et al. (2009), multiple neural networks were logically integrated, where each network was trained on only a part of the input space, and the output was integrated at the prediction phase by selecting the network that corresponded to the input subspace. This is a competitive integration type scheme. A more complex integration was done in de Nardi et al. (2006), where neural network components for a helicopter control system were cooperatively integrated based on the AL of a hand designed PID controller. In Wang et al. (2012), two CNNs, one being a text detector and the other a character recognizer, were logically integrated, where the detector determined image locations containing text and the recognizer extracted text given these locations. In Eppel (2017), the recognition of the parts of an object was done in two steps, where in the first step, a CNN was used to segment the image to separate the object from its background, then in the second step another CNN was applied to the original image and the segmentation map to identify the object parts.

3.4.2 Learned

Learned integration consists of the composition of modules through a learning algorithm. Here, learning is concerned with how to optimally combine modules in order to obtain the best possible performance on the target problem. Composing modules to solve a certain problem is not straightforward, involving complex interactions between modules. Using learning algorithms in modular integration helps to capture useful complex relationships between modules. Even when subproblems are readily composable into a final solution, learning algorithms can find shortcuts that can help formulate more efficient solutions. However, the introduction of learning can result in unnecessary computational overhead, and can give rise to tightly coupled modules, often leading to overfitting and harder debugging. A very common type of learned integration is synaptic integration, where different modules are combined together by converging to a common parametric layer, which determines, through learning, the contribution of each module to the final output.

In Almasri and Kaluarachchi (2005), several neural network outputs were integrated together, to predict nitrate distribution in ground water, using a gating network. A gating network is a very common integration technique, where a specialised network is used to predict a vector of weights, which is used to combine the outputs of different experts (i.e networks). In Zheng et al. (2006), a Bayesian probability model was used to combine MLP and RBF network predictions based on how well each module performed in previous examples. This Bayesian model tended to give more weight to the module that performed better on previous examples in a certain target period of prediction. Fuzzy logic has also been used as a tool for learned integration (Melin et al. 2007; Hidalgo et al. 2009). In Mendoza et al. (2009a, b) and Melin et al. (2011) an image recognition MNN was proposed where different neural networks were trained on part of the edge vector extracted from the target image. Fuzzy logic was then used to integrate different neural network outputs by assessing the relevance of each module using a fuzzy inference system and integrating using the Sugeno integral. Synaptic integration was done in Anderson et al. (2016), with the aim of achieving transfer learning on a small amount of data. A set of untrained modules were added to a pretrained network and integrated by learning while freezing the original network weights. In another work (Terekhov et al. 2015), a similar approach utilized synaptic integration for multi-task learning, where an initial network was trained on some task, followed by a modular block of neurons that were added and integrated by training on a different task, while again freezing the original network parameters.

In the next section, we apply the discussed modular framework to some of the state-of-the-art MNNs, where we analyse their modular composition and show how their general design is captured by our abstracted modular concepts, in a proof-of-concept of the practical applicability of these modularization principles.

4 Case studies

We will present some case studies of state-of-the-art MNNs, for which we will emphasize the modular techniques applied. We will be concerned with the three main levels of modularization, i.e. topology, formation and integration. As discussed earlier, domain modularization is an optional component. In the Inception architecture (Szegedy et al. 2015), the architecture which set the state-of-the-art in ILSVRC14, the topology is mainly a repeated block architecture, which combines both multi-path and sequential structures. The main skeleton is a sequence of repeated blocks, where each block is a multi-path subnetwork. Formation was manually engineered. The main guiding heuristic for the manual formation was approximating a sparse architecture using dense modules, such that it can still exploit the hardware optimized for dense computations. Integration is a learned integration.

FractalNet (Larsson et al. 2016) topology is a combination of different modular techniques. The main skeleton is a sequential repeated block structure. Each block is a combination of recursive fractal structure, sequential chaining and multi-path branching. The manual formation is based on making the network robust to the choice of overall depth, through the recursive nesting of subnetworks of various depths, such that the learning algorithm can carve out the efficient paths. Integration is a learned integration.

CapsNet (Sabour et al. 2017) can be considered a modular node topology, where each single activation is replaced by a pose vector capturing different instantiation parameters of the underlying object. Formation is manually engineered, where the main guiding principle is to achieve equivariance through pose vectors, transformation matrices and dynamic routing by agreement. Integration is Arithmetic-Logic based on dynamic routing by agreement. Dynamic routing by agreement acts by combining predictions of the lower layer based on their relative agreement. In that way, it has no learned state.

PCNet++ (Phaye et al. 2018), a variant of CapsNet (Sabour et al. 2017) has a topology of a sequential and mutli-architectural nature. The main skeleton is a sequential stack of three simpler networks called DCNet. The output of each subnetwork is routed into two branches, one contributes to the final output and the other is used as the input to the next subnetwork in the stack. This way, each subnetwork builds on the previous one and at the same time contributes to the final output as a standalone architecture. The formation is manual based on diversifying the PrimaryCapsules, such that different capsules carry information of various scales of the image. Integration is a simple Arithmetic-Logic, where the DigitsCapsules from different output levels are concatenated to form the final output.

In their neural architecture search (NAS) technique, Liu et al. (2017) used a search space of hierarchical modular topology, a special case of modular node topology as discussed earlier. Each searched architecture is a hierarchical structure of building blocks of increasing complexity starting from a pre-defined set of primitives. The techniques for formation and integration were evolutionary search with tournament selection and learned integration, respectively. Another NAS technique was introduced by Bender et al. (2018). This search technique is based on training an exponential number of modular architectures through training a large model, called one-shot model, that includes these architectures as subnetworks. The main topological structure of the one-shot model is composed of a sequential skeleton of multi-path blocks. Formation is a hybrid of implicit and explicit learned formation, where a path-dropping technique is used for regularization such that the one-shot model can be used as a proxy for the performance measure of the different implicit subnetworks. After convergence of the one-shot model, explicit modular networks are sampled based on the performance measure provided by the one-shot model. Integration is a learned integration.

5 Conclusion

This review aimed at introducing and analysing the main modularization techniques used in the field of neural networks so far, in an attempt to provide researchers and practitioners with insights on how to systematically implement neural modularity in order to harness its advantages. We devised a taxonomy of modularization techniques with four main categories, based on the target of the modularization process, i.e.: domain, topology, formation and integration. We further divided each category into subcategories based on the nature of the techniques involved. We discussed the advantages and disadvantages of the techniques, and how they are used to solve real-world problems. Analysis and empirical results show that modularity can be advantageous over monolithic networks in many situations.

The review has shown that a wide variety of algorithms for modularization exists, acting on different parts of the MNN life cycle. We have shown that advances in MNNs are not restricted to biologically inspired topological modularity. The quest for modularity in ANNs is far from being a mere case of enforcing networks to be partial replicas of the brain. Even topological modularity is often a vague imitation of brain structure. As the ANN literature has increasingly diverged from its early biological roots, so has modularity metamorphosed into different shapes and techniques, ranging from biologically-inspired to purely engineering practices.

The techniques reviewed here have ranged from explicit expert-knowledge based techniques to fully automated implicit modularization techniques, each having its specific set of pros and cons and suitability for particular problems. Some techniques were found to be tailored to satisfy the specific constraints of particular problems, while others were found to be generic, trading specialization performance for full automation and generalizability. Neural modularization was shown to be a sequential application of techniques, which we called modularization chain, where each technique acts on a different aspect of the neural network.

Although, as discussed, modularity has many advantages over monolithic deep networks, the main trend is still oriented towards monolithic deep neural networks. This is mainly due to the many successes of monolithic deep learning in different areas throughout the last decade. Also, the lack of extensive research into learning and formation techniques for neural modularity makes it hard for practitioners to efficiently deploy the approach. Contrary to this, monolithic networks have attracted extensive research that has generated a critical mass of theoretical insights and practical tricks, which facilitate their deployment. Evolutionary algorithms are currently the main actors in complex modular neural network construction. However, the debate of whether evolutionary algorithms are the best approach for MNN formation and if they harness the full power of modularization and problem decomposition is still open. Also, there is still a significant gap on how to stimulate problem decomposition in modular networks, so that their topological modularity may also become a full functional modularity.

We tentatively predict that as the challenges facing deep learning become increasingly hard, a saturation phase will eventually be reached where depth and learning tricks may not be enough to fuel progress in deep learning. We dont view modularity as a replacement for depth, but as a complementary and integrable approach to deep learning, especially given that excessive depth is becoming increasingly criticized for reasons of computational cost and extraneousness. The dilemma is similar to the software quality problem, where exponential growth in hardware efficiency is masking poor algorithmic optimization. We believe that as deep learning becomes increasingly applied to more challenging and general problems, the need for robust Artificial General Intelligence practices will sooner or later promote the modularization of neural networks.

References

Achard S, Bullmore E (2007) Efficiency and cost of economical brain functional networks. PLoS Comput Biol 3(2):0174–0183. https://doi.org/10.1371/journal.pcbi.0030017

Aguirre C, Huerta R, Corbacho F, Pascual P (2002) Analysis of biologically inspired small-world networks. In: International conference on artificial neural networks. Springer, pp 27–32

Allen F, Almasi G, Andreoni W, Beece D, Berne BJ, Bright A, Brunheroto J, Cascaval C, Castanos J, Coteus P, Crumley P, Curioni A, Denneau M, Donath W, Eleftheriou M, Flitch B, Fleischer B, Georgiou CJ, Germain R, Giampapa M, Gresh D, Gupta M, Haring R, Ho H, Hochschild P, Hummel S, Jonas T, Lieber D, Martyna G, Maturu K, Moreira J, Newns D, Newton M, Philhower R, Picunko T, Pitera J, Pitman M, Rand R, Royyuru A, Salapura V, Sanomiya A, Shah R, Sham Y, Singh S, Snir M, Suits F, Swetz R, Swope WC, Vishnumurthy N, Ward TJC, Warren H, Zhou R (2001) Blue Gene: a vision for protein science using a petaflop supercomputer. IBM Syst J 40(2):310–327. https://doi.org/10.1147/sj.402.0310

Almasri MN, Kaluarachchi JJ (2005) Modular neural networks to predict the nitrate distribution in ground water using the on-ground nitrogen loading and recharge data. Environ Model Softw 20(7):851–871

Aminian M, Aminian F (2007) A modular fault-diagnostic system for analog electronic circuits using neural networks with wavelet transform as a preprocessor. IEEE Trans Instrum Meas 56(5):1546–1554

Anand R, Mehrotra K, Mohan C, Ranka S (1995) Efficient classification for multiclass problems using modular neural networks. IEEE Trans Neural Netw 6(1):117–124. https://doi.org/10.1109/72.363444

Anderson A, Shaffer K, Yankov A, Corley CD, Hodas NO (2016) Beyond fine tuning: a modular approach to learning on small data. arXiv:1611.01714v1

Andreas J, Rohrbach M, Darrell T, Klein D (2016a) Neural module networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp 39–48

Andreas J, Rohrbach M, Darrell T, Klein D (2016b) Learning to compose neural networks for question answering. arXiv:1601.01705

Angelucci a, Clascá F, Bricolo E, Cramer KS, Sur M (1997) Experimentally induced retinal projections to the ferret auditory thalamus: development of clustered eye-specific patterns in a novel target. J Neurosci Off J Soc Neurosci 17(6):2040–2055

Auda G, Kamel M (1998) Modular neural network classifiers: a comparative study. J Intell Robot Syst 21:117–129. https://doi.org/10.1023/A:1007925203918

Auda G, Kamel M (1999) Modular neural networks: a survey. Int J Neural Syst 9(2):129–51

Azam F (2000) Biologically inspired modular neural networks. https://vtechworks.lib.vt.edu/handle/10919/27998. Accessed 23 Dec 2018

Ba J, Caruana R (2014) Do deep nets really need to be deep? In: Advances in neural information processing systems. pp 2654–2662

Babaei S, Geranmayeh A, Seyyedsalehi SA (2010) Protein secondary structure prediction using modular reciprocal bidirectional recurrent neural networks. Comput Methods Programs Biomed 100(3):237–247. https://doi.org/10.1016/j.cmpb.2010.04.005

Battaglia PW, Hamrick JB, Bapst V, Sanchez-Gonzalez A, Zambaldi V, Malinowski M, Tacchetti A, Raposo D, Santoro A, Faulkner R, Gulcehre C, Song F, Ballard A, Gilmer J, Dahl G, Vaswani A, Allen K, Nash C, Langston V, Dyer C, Heess N, Wierstra D, Kohli P, Botvinick M, Vinyals O, Li Y, Pascanu R (2018) Relational inductive biases, deep learning, and graph networks. arXiv:1806.01261

Bender G, Kindermans PJ, Zoph B, Vasudevan V, Le Q (2018) Understanding and simplifying one-shot architecture search. http://proceedings.mlr.press/v80/bender18a. Accessed 5 Dec 2018

Bengio Y, Louradour J, Collobert R, Weston J (2009) Curriculum learning. In: Proceedings of the 26th annual international conference on machine learning—ICML '09. ACM Press, New York, New York, USA, pp 1–8. https://doi.org/10.1145/1553374.1553380, http://portal.acm.org/citation.cfm?doid=1553374.1553380

Bengio S, Vinyals O, Jaitly N, Shazeer N (2015) Scheduled sampling for sequence prediction with recurrent neural networks. http://papers.nips.cc/paper/5956-scheduled-sampling-for-sequence-prediction-with-recurrent-neural-networks. Accessed 12 Mar 2018

Bhende C, Mishra S, Panigrahi B (2008) Detection and classification of power quality disturbances using S-transform and modular neural network. Electr Power Syst Res 78(1):122–128. https://doi.org/10.1016/j.epsr.2006.12.011

Blundell C, Cornebise J, Kavukcuoglu K, Wierstra D (2015) Weight uncertainty in neural networks. arXiv preprint arXiv:1505.05424

Bohland JW, Minai AA (2001) Efficient associative memory using small-world architecture. Neurocomputing 38:489–496. https://doi.org/10.1016/S0925-2312(01)00378-2

Brandes U, Delling D, Gaertler M, Gorke R, Hoefer M, Nikoloski Z, Wagner D (2008) On modularity clustering. IEEE Trans Knowl Data Eng 20(2):172–188. https://doi.org/10.1109/TKDE.2007.190689

Braylan A, Hollenbeck M, Meyerson E, Miikkulainen R (2015) Reuse of neural modules for general video game playing. arXiv:1512.01537

Bullmore ET, Bassett DS (2011) Brain graphs: graphical models of the human brain connectome. Annu Rev Clin Psychol 7(1):113–140. https://doi.org/10.1146/annurev-clinpsy-040510-143934

Bullmore E, Sporns O (2009) Complex brain networks: graph theoretical analysis of structural and functional systems. Nat Rev Neurosci 10(3):186–198. https://doi.org/10.1038/nrn2575

Buxhoeveden DP (2002) The minicolumn hypothesis in neuroscience. Brain 125(5):935–951. https://doi.org/10.1093/brain/awf110

Caelli T, Guan L, Wen W (1999) Modularity in neural computing. Proc IEEE 87(9):1497–1518. https://doi.org/10.1109/5.784227

Calabretta R, Nolfi S, Parisi D, Wagner GP (2000) Duplication of modules facilitates the evolution of functional specialization. Artif Life 6(1):69–84

Chen ZJ, He Y, Rosa-Neto P, Germann J, Evans AC (2008) Revealing modular architecture of human brain structural networks by using cortical thickness from MRI. Cereb Cortex 18(10):2374–2381. https://doi.org/10.1093/cercor/bhn003

Chiang CC, Fu HC (1994) A divide-and-conquer methodology for modular supervised neural network design. In: Neural networks, 1994. IEEE world congress on computational intelligence, 1994 IEEE international conference on. IEEE, vol 1, pp 119–124

Chihaoui M, Elkefi A, Bellil W, Ben Amar C (2016) A survey of 2D face recognition techniques. Computers 5(4):21. https://doi.org/10.3390/computers5040021

Chollet F (2016) Xception: deep learning with depthwise separable convolutions. arXiv:1610.02357

Chris Tseng H, Almogahed B (2009) Modular neural networks with applications to pattern profiling problems. Neurocomputing 72(10–12):2093–2100. https://doi.org/10.1016/J.NEUCOM.2008.10.020

Ciregan D, Meier U, Schmidhuber J (2012) Multi-column deep neural networks for image classification. In: Computer vision and pattern recognition (CVPR), 2012 IEEE conference on. IEEE, pp 3642–3649

Clune J, Mouret JB, Lipson H (2013) The evolutionary origins of modularity. Proc Biol Sci R Soc 280(1755):20122863. https://doi.org/10.1098/rspb.2012.2863. arXiv:1207.2743v1

de Nardi R, Togelius J, Holland O, Lucas S (2006) Evolution of neural networks for helicopter control: Why modularity matters. In: 2006 IEEE international conference on evolutionary computation. IEEE, pp 1799–1806. https://doi.org/10.1109/CEC.2006.1688525

Di Ferdinando A, Calabretta R, Parisi D (2001) Evolving modular architectures for neural networks. Proc Sixth Neural Comput Psychol Workshop Evol Learn Dev 12(5):253–262

Douglas RJ, Martin KAC (2007) Recurrent neuronal circuits in the neocortex. Curr Biol CB 17(13):R496–500. https://doi.org/10.1016/j.cub.2007.04.024

Eppel S (2017) Hierarchical semantic segmentation using modular convolutional neural networks. arXiv:1710.05126v1

Eyben F, Weninger F, Squartini S, Schuller B (2013) Real-life voice activity detection with LSTM recurrent neural networks and an application to Hollywood movies. In: ICASSP, IEEE international conference on acoustics, speech and signal processing—proceedings, pp 483–487. https://doi.org/10.1109/ICASSP.2013.6637694

Fernando C, Banarse D, Blundell C, Zwols Y, Ha D, Rusu AA, Pritzel A, Wierstra D (2017) PathNet: evolution channels gradient descent in super neural networks. arXiv:1701.08734

Ferreira MD, Corrêa DC, Nonato LG, de Mello RF (2018) Designing architectures of convolutional neural networks to solve practical problems. Expert Syst Appl 94:205–217. https://doi.org/10.1016/J.ESWA.2017.10.052

Franco L, Cannas SA (2001) Generalization properties of modular networks: implementing the parity function. IEEE Trans Neural Netw 12(6):1306–1313. https://doi.org/10.1109/72.963767

Freddolino PL, Liu F, Gruebele M, Schulten K (2008) Ten-microsecond molecular dynamics simulation of a fast-folding WW domain. Biophys J 94(10):L75–L77. https://doi.org/10.1529/biophysj.108.131565

Fritsch J (1996) Modular neural networks for speech recognition (No. CMU-CS-96-203). Carnegie-Mellon Univ Pittsburgh PA Dept of Computer Science

Fu HC, Lee YP, Chiang CC, Pao HT (2001) Divide-and-conquer learning and modular perceptron networks. IEEE Trans Neural Netw 12(2):250–263. https://doi.org/10.1109/72.914522

Fukushima K, Miyake S, Ito T (1983) Neocognitron: a neural network model for a mechanism of visual pattern recognition. IEEE Trans Syst Man Cybern SMC–13(5):826–834. https://doi.org/10.1109/TSMC.1983.6313076

Garcia-Pedrajas N, Hervas-Martinez C, Munoz-Perez J (2003) COVNET: a cooperative coevolutionary model for evolving artificial neural networks. IEEE Trans Neural Netw 14(3):575–596. https://doi.org/10.1109/TNN.2003.810618

Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE (2017) Neural message passing for quantum chemistry. arXiv:1704.01212

Gollisch T, Meister M (2010) Eye smarter than scientists believed: neural computations in circuits of the retina. https://doi.org/10.1016/j.neuron.2009.12.009

Goltsev A, Gritsenko V (2015) Modular neural networks with radial neural columnar architecture. Biol Inspir Cognit Archit 13:63–74. https://doi.org/10.1016/J.BICA.2015.06.001

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. http://papers.nips.cc/paper/5423-generative-adversarial-nets. Accessed 23 Dec 2018

Gradojevic N, Gençay R, Kukolj D (2009) Option pricing with modular neural networks. IEEE Trans Neural Netw Publ IEEE Neural Netw Council 20(4):626–637. https://doi.org/10.1109/TNN.2008.2011130

Guan SU, Li S (2002) Parallel growing and training of neural networks using output parallelism. IEEE Trans Neural Netw 13(3):542–550

Happel BLM, Murre JMJ (1994) Design and evolution of modular neural network architectures. Neural Netw 7(6–7):985–1004. https://doi.org/10.1016/S0893-6080(05)80155-8

Haykin S (1994) Neural networks: a comprehensive foundation. Prentice Hall PTR, Upper Saddle River

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp 770–778

Hidalgo D, Castillo O, Melin P (2009) Type-1 and type-2 fuzzy inference systems as integration methods in modular neural networks for multimodal biometry and its optimization with genetic algorithms. Inf Sci 179(13):2123–2145

Hochreiter S, Urgen Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hu R, Rohrbach M, Andreas J, Darrell T, Saenko K (2016) Modeling relationships in referential expressions with compositional modular networks. arXiv:1611.09978

Huang G-B (2003) Learning capability and storage capacity of two-hidden-layer feedforward networks. IEEE Trans Neural Netw 14(2):274–281. https://doi.org/10.1109/TNN.2003.809401

Huang G, Sun Y, Liu Z, Sedra D, Weinberger KQ (2016) Deep networks with stochastic depth. In: European conference on computer vision. Springer, pp 646–661

Huizinga J, Mouret JB, Clune J (2014) Evolving neural networks that are both modular and regular: HyperNeat plus the connection cost technique. Gecco, pp 697–704, https://doi.org/10.1145/2576768.2598232

Hüsken M, Igel C, Toussaint M (2002) Task-dependent evolution of modularity in neural networks. Connect Sci 14(3):219–229

Jacobs RA, Jordan MI, Nowlan SJ, Hinton GE (1991) Adaptive mixtures of local experts. Neural Comput 3(1):79–87. https://doi.org/10.1162/neco.1991.3.1.79

Wei Jiang, Kong Seong G (2007) Block-based neural networks for personalized ECG signal classification. IEEE Trans Neural Netw 18(6):1750–1761. https://doi.org/10.1109/TNN.2007.900239

Kacprzyk J, Pedrycz W (2015) Springer handbook of computational intelligence. Springer, Berlin

Kaiser M, Hilgetag CC (2010) Optimal hierarchical modular topologies for producing limited sustained activation of neural networks. Front Neuroinform 4:8

Karami M, Safabakhsh R, Rahmati M (2013) Modular cellular neural network structure for wave-computing-based image processing. ETRI J 35(2):207–217. https://doi.org/10.4218/etrij.13.0112.0107

Kashtan N, Alon U (2005) Spontaneous evolution of modularity and network motifs. Proc Natl Acad Sci USA 102(39):13773–8. https://doi.org/10.1073/pnas.0503610102

Kastellakis G, Cai DJ, Mednick SC, Silva AJ, Poirazi P (2015) Synaptic clustering within dendrites: an emerging theory of memory formation. https://doi.org/10.1016/j.pneurobio.2014.12.002

Kim T, Cha M, Kim H, Lee JK, Kim J (2017) Learning to discover cross-domain relations with generative adversarial networks. arXiv:1703.05192

Larsson G, Maire M, Shakhnarovich G (2016) FractalNet: ultra-deep neural networks without residuals. arXiv:1605.07648

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

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436

Lin M, Chen Q, Yan S (2013) Network in network. arXiv preprint p 10, https://doi.org/10.1109/ASRU.2015.7404828, arXiv:1312.4400

Liu H, Simonyan K, Vinyals O, Fernando C, Kavukcuoglu K (2017) Hierarchical representations for efficient architecture search. arXiv:1711.00436

Lodato S, Arlotta P (2015) Generating neuronal diversity in the mammalian cerebral cortex. Annu Rev Cell Dev Biol 31(1):699–720. https://doi.org/10.1146/annurev-cellbio-100814-125353

López-Muñoz F, Boya J, Alamo C (2006) Neuron theory, the cornerstone of neuroscience, on the centenary of the Nobel Prize award to Santiago Ramón y Cajal. Brain Res Bull 70(4–6):391–405. https://doi.org/10.1016/j.brainresbull.2006.07.010

Melin P, Mancilla A, Lopez M, Mendoza O (2007) A hybrid modular neural network architecture with fuzzy sugeno integration for time series forecasting. Appl Soft Comput 7(4):1217–1226

Melin P, Mendoza O, Castillo O (2011) Face recognition with an improved interval type-2 fuzzy logic sugeno integral and modular neural networks. IEEE Trans Syst Man Cybern Part A Syst Hum 41(5):1001–1012

Mendoza O, Melin P, Licea G (2009a) A hybrid approach for image recognition combining type-2 fuzzy logic, modular neural networks and the Sugeno integral. Inf Sci 179(13):2078–2101. https://doi.org/10.1016/j.ins.2008.11.018

Mendoza O, Melín P, Castillo O (2009b) Interval type-2 fuzzy logic and modular neural networks for face recognition applications. Appl Soft Comput 9(4):1377–1387. https://doi.org/10.1016/j.asoc.2009.06.007

Meunier D, Lambiotte R, Bullmore ET (2010) Modular and hierarchically modular organization of brain networks. Front Neurosci. https://doi.org/10.3389/fnins.2010.00200

Miikkulainen R, Liang J, Meyerson E, Rawal A, Fink D, Francon O, Raju B, Shahrzad H, Navruzyan A, Duffy N, Hodjat B (2017) Evolving deep neural networks. arXiv:1703.00548

Montufar GF, Pascanu R, Cho K, Bengio Y (2014) On the number of linear regions of deep neural networks. http://papers.nips.cc/paper/5422-on-the-number-of-linear-regions-of-deep-neural-networks. Accessed 24 Dec 2018

Moon S-W, Kong S-G (2001) Block-based neural networks. IEEE Trans Neural Netw 12(2):307–317. https://doi.org/10.1109/72.914525

Mountcastle VB (1997) The columnar organization of the neocortex. Brain J Neurol. https://doi.org/10.1093/brain/120.4.701

Mouret JB, Doncieux S (2009) Evolving modular neural-networks through exaptation. In: 2009 IEEE congress on evolutionary computation, CEC 2009. pp 1570–1577. https://doi.org/10.1109/CEC.2009.4983129

Mouret JB, Doncieux S (2008) MENNAG: a modular, regular and hierarchical encoding for neural-networks based on attribute grammars. Evolut Intell 1(3):187–207. https://doi.org/10.1007/s12065-008-0015-7

Newman MEJ (2004) Detecting community structure in networks. Eur Phys J B 38:321–330. https://doi.org/10.1140/epjb/e2004-00124-y

Newman MEJ (2006) Modularity and community structure in networks. Proc Natl Acad Sci USA 103(23):8577–82. https://doi.org/10.1073/pnas.0601602103

Newman MEJ (2016) Community detection in networks: modularity optimization and maximum likelihood are equivalent. 1:1–8. https://doi.org/10.1103/PhysRevE.94.052315, arXiv:1606.02319

Oh IS, Suen CY (2002) A class-modular feedforward neural network for handwriting recognition. Pattern Recognit 35(1):229–244. https://doi.org/10.1016/S0031-3203(00)00181-3

Ortín S, Gutiérrez J, Pesquera L, Vasquez H (2005) Nonlinear dynamics extraction for time-delay systems using modular neural networks synchronization and prediction. Physica A Stat Mech Appl 351(1):133–141. https://doi.org/10.1016/j.physa.2004.12.015

Ou G, Murphey YL (2007) Multi-class pattern classification using neural networks. Pattern Recognit 40(1):4–18. https://doi.org/10.1016/j.patcog.2006.04.041

Pan P, Xu Z, Yang Y, Wu F, Zhuang Y (2016) Hierarchical recurrent neural encoder for video representation with application to captioning. In: The IEEE conference on computer vision and pattern recognition (CVPR)

Phan KT, Maul TH, Tuong TV (2015) A parallel circuit approach for improving the speed and generalization properties of neural networks. In: 2015 11th international conference on natural computation (ICNC). IEEE, pp 1–7. https://doi.org/10.1109/ICNC.2015.7377956

Phan KT, Maul TH, Vu TT, Lai WK (2016) Improving neural network generalization by combining parallel circuits with dropout. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), vol 9949 LNCS, pp 572–580. https://doi.org/10.1007/978-3-319-46675-0_63, arXiv:1612.04970

Phan KT, Maul TH, Vu TT, Lai WK (2017) Dropcircuit: a modular regularizer for parallel circuit networks. Neural Process Lett 47:1–18

Phaye SSR, Sikka A, Dhall A, Bathula D (2018) Dense and diverse capsule networks: making the capsules learn better. arXiv:1805.04001

Radicchi F, Castellano C, Cecconi F, Loreto V, Parisi D (2004) Defining and identifying communities in networks. Proc Natl Acad Sci USA 101(9):2658–2663

Reisinger J, Stanley KO, Miikkulainen R (2004) Evolving reusable neural modules. In: Genetic and evolutionary computation conference. Springer, pp 69–81

Ronco E, Gawthrop P (1995) Modular neural networks: a state of the art. Rapport Technique CSC-95026, center of system and control, University of Glasgow. http://www.mech.gla.ac.uk/control/report.html

Ronen M, Shabtai Y, Guterman H (2002) Hybrid model building methodology using unsupervised fuzzy clustering and supervised neural networks. Biotechnol Bioeng 77(4):420–429

Rudasi L, Zahorian S (1991) Text-independent talker identification with neural networks. In: [Proceedings] ICASSP 91: 1991 international conference on acoustics, speech, and signal processing. IEEE, vol 1, pp 389–392. https://doi.org/10.1109/ICASSP.1991.150358

Sabour S, Frosst N, Hinton GE (2017) Dynamic routing between capsules. http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules. Accessed 28 Feb 2018

San PP, Ling SH, Nguyen HT (2011) Block based neural network for hypoglycemia detection. In: 2011 annual international conference of the IEEE engineering in medicine and biology society. IEEE, pp 5666–5669. https://doi.org/10.1109/IEMBS.2011.6091371

Santoro A, Raposo D, Barrett DGT, Malinowski M, Pascanu R, Battaglia P, Lillicrap T (2017) A simple neural network module for relational reasoning. arXiv:1706.01427

Schwarz AJ, Gozzi A, Bifone A (2008) Community structure and modularity in networks of correlated brain activity. Magn Reson Imaging 26(7):914–920. https://doi.org/10.1016/j.mri.2008.01.048

Serban IV, Sordoni A, Bengio Y, Courville A, Pineau J (2016) Building end-to-end dialogue systems using generative hierarchical neural network models. AAAI p 8. https://doi.org/10.1017/CBO9781107415324.004, arXiv:1507.04808

Sharkey AJC (1996) On combining artificial neural nets. Connect Sci 8(3–4):299–313. https://doi.org/10.1080/095400996116785

Shetty R, Laaksonen J (2015) Video captioning with recurrent networks based on frame- and video-level features and visual content classification. arXiv:1512.02949

Singh S, Hoiem D, Forsyth D (2016) Swapout: Learning an ensemble of deep architectures. In: Advances in neural information processing systems. pp 28–36

Song L, Zhang Y, Wang Z, Gildea D (2018) A graph-to-sequence model for AMR-to-text generation. arXiv:1805.02473
Soutner D, Müller L (2013) Application of lstm neural networks in language modelling. In: International conference on text, speech and dialogue. Springer, pp 105–112
Sporns O (2011) The human connectome: a complex network. Ann N Y Acad Sci. https://doi.org/10.1111/j.1749-6632.2010.05888.x
Sporns O, Zwi JD (2004) The small world of the cerebral cortex. Neuroinformatics 2(2):145–162. https://doi.org/10.1385/NI:2:2:145
Srivastava RK, Masci J, Kazerounian S, Gomez F, Schmidhuber J (2013) Compete to compute. Nips pp 2310–2318
Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15:1929–1958. https://doi.org/10.1214/12-AOS1000. arXiv:1102.4807
Srivastava RK, Greff K, Schmidhuber J (2015) Highway networks. arXiv:1505.00387 [cs]
Stanley KO, Miikkulainen R (2002) Evolving neural networks through augmenting topologies. Evolut Comput 10(2):99–127. https://doi.org/10.1162/106365602320169811
Stanley KO, D’Ambrosio DB, Gauci J (2009) A hypercube-based encoding for evolving large-scale neural networks. Artif Life 15(2):185–212. https://doi.org/10.1162/artl.2009.15.2.15202
Stollenga MF, Byeon W, Liwicki M, Schmidhuber J (2015) Parallel multi-dimensional LSTM, with application to fast biomedical volumetric image segmentation. In: Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R (eds) Advances in neural information processing systems, vol 28. Curran Associates Inc, Red Hook, pp 2998–3006
Subirats JL, Jerez JM, Gómez I, Franco L (2010) Multiclass pattern recognition extension for the new C-Mantec constructive neural network algorithm. Cognit Comput 2(4):285–290. https://doi.org/10.1007/s12559-010-9051-6
Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition. vol 07-12-June, pp 1–9. https://doi.org/10.1109/CVPR.2015.7298594, arXiv:1409.4842
Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp 2818–2826
Terekhov AV, Montone G, O’Regan JK (2015) Knowledge transfer in deep block-modular neural networks. Springer, Cham, pp 268–279. https://doi.org/10.1007/978-3-319-22979-9_27
Tyler JR, Wilkinson DM, Huberman BA (2005) E-Mail as spectroscopy: automated discovery of community structure within organizations. Inf Soc 21(2):143–153. https://doi.org/10.1080/01972240590925348
Veit A, Wilber MJ, Belongie S (2016) Residual networks behave like ensembles of relatively shallow networks. In: Advances in neural information processing systems. pp 550–558
Verbancsics P, Stanley KO (2011) Constraining connectivity to encourage modularity in HyperNEAT. In: Proceedings of the 13th annual conference on Genetic and evolutionary computation—GECCO ’11. p 1483. https://doi.org/10.1145/2001576.2001776
Vlahogianni EI, Karlaftis MG, Golias JC (2007) Spatio-temporal short-term urban traffic volume forecasting using genetically optimized modular networks. Comput Aided Civ Infrastruct Eng 22(5):317–325
Waibel A (1989) Modular construction of time-delay neural networks for speech recognition. Neural Comput 1(1):39–46. https://doi.org/10.1162/neco.1989.1.1.39
Wang M (2015) Multi-path convolutional neural networks for complex image classification. arXiv:1506.04701
Wang SJ, Hilgetag CC, Zhou C (2011) Sustained activity in hierarchical modular neural networks: self-organized criticality and oscillations. Front Comput Neurosci 5:30
Wang T, Wu DJ, Coates A, Ng AY (2012) End-to-end text recognition with convolutional neural networks. In: Pattern recognition (ICPR), 2012 21st international conference on. IEEE, pp 3304–3308
Watanabe C, Hiramatsu K, Kashino K (2018) Modular representation of layered neural networks. Neural Netw 97:62–73. https://doi.org/10.1016/J.NEUNET.2017.09.017
Watts DJ (1999) Networks, dynamics, and the smallworld phenomenon. Am J Sociol 105(2):493–527. https://doi.org/10.1086/210318
Weston J, Chopra S, Bordes A (2014) Memory networks. arXiv:1410.3916
Xie S, Girshick R, Dollár P, Tu Z, He K (2016) Aggregated residual transformations for deep neural networks. arXiv preprint arXiv:1611.05431
Xu L, Krzyzak A, Suen C (1992) Methods of combining multiple classifiers and their applications to handwriting recognition. IEEE Trans Syst Man Cybern 22(3):418–435. https://doi.org/10.1109/21.155943
Yu L, Lin Z, Shen X, Yang J, Lu X, Bansal M, Berg TL (2018) MAttNet: modular attention network for referring expression comprehension. arXiv:1801.08186v2

Yu H, Wang J, Huang Z, Yang Y, Xu W (2016) Video paragraph captioning using hierarchical recurrent neural networks. In: The IEEE conference on computer vision and pattern recognition (CVPR)

Yuan M, Lin Y (2006) Model selection and estimation in regression with grouped variables. J R Stat Soc Ser B (Stat Methodol) 68(1):49–67. https://doi.org/10.1111/J.1467-9868.2005.00532.X@10.1111/(ISSN)1467-9868.TOP_SERIES_B_RESEARCH, https://rss.onlinelibrary.wiley.com/doi/full/10.1111/j.1467-9868.2005.00532.x%4010.1111/%28ISSN%291467-9868.TOP_SERIES_B_RESEARCH

Zhang N, Donahue J, Girshick R, Darrell T (2014) Part-based R-CNNs for fine-grained category detection. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics). LNCS, vol 8689 pp 834–849. https://doi.org/10.1007/978-3-319-10590-1_54, arXiv:1407.3867

Zhang F, Leitner J, Milford M, Corke P (2016) Modular deep Q networks for sim-to-real transfer of visuo-motor policies. arXiv:1610.06781v4

Zhang C, Ren M, Urtasun R (2018) Graph HyperNetworks for neural architecture search. arXiv:1810.05749

Zheng W, Lee DH, Shi Q (2006) Short-term freeway traffic flow prediction: Bayesian combined neural network approach. J Transp Eng 132(2):114–121. https://doi.org/10.1061/(ASCE)0733-947X(2006)132:2(114)