Structural Learning in Artificial Neural Networks: A Neural Operator Perspective

Anonymous authors
Paper under double-blind review

Abstract

Over the history of Artificial Neural Networks (ANNs), only a minority of algorithms integrate structural changes of the network architecture into the learning process. Modern neuroscience has demonstrated that biological learning is largely structural, with mechanisms such as synaptogenesis and neurogenesis present in adult brains and considered important for learning. Despite this history of artificial methods and biological inspiration, and furthermore the recent resurgence of neural methods in deep learning, relatively few current ANN methods include structural changes in learning compared to those that only adjust synaptic weights during the training process. We aim to draw connections between different approaches of structural learning that have similar abstractions in order to encourage collaboration and development. In this review, we provide a survey on structural learning methods in deep ANNs, including a new neural operator framework from a cellular neuroscience context and perspective aimed at motivating research on this challenging topic. We first give a background on biological developmental processes in the brain. We then provide an overview of ANN methods which include structural changes within the neural operator framework in the learning process, diving into each neural operator in detail. Finally, we present overarching trends in how these operators are implemented and discuss the open challenges in structural learning in ANNs.

1 Introduction

Artificial intelligence algorithms, and specifically artificial neural networks (ANNs), have been able to grow exponentially in power and complexity due to new automated techniques such as backpropagation that replace some of what has been previously hand designed, as well as breakthroughs in computational power. However, many assumptions of current learning approaches such as structurally static learning and isolated training from scratch have diverged from what is observed in the brain, the original inspiration for neural networks. While such assumptions have thus far helped push the frontier of artificial intelligence to its current state by adapting to the characteristics of artificial hardware and software, they may become a bottleneck preventing more powerful ANNs that don't require hand-designed architectures.

Although ANNs with the simplest architecture of an arbitrarily wide single hidden layer are theoretically capable of approximating any function (Cybenko, 1989; Hornik, 1991), deep ANNs with more nuanced structures have taken over in popularity in practice. Computational capacities have improved to allow such large and deep networks. An ANN structured specifically for a given domain can approximate a function more practically and efficiently in storage and training time than a generic wide shallow network (Mhaskar & Poggio, 2016; Poggio et al., 2017). Even deep dense ANNs use structural priors and are not completely dense, as each neuron is only connected to the neurons in the immediately preceding and following layers, rather than all neurons in a network. This imposes a small amount of structure on the network. Convolutional networks are even sparser than this, limiting neurons to have only spatially local connections and imposing weight-sharing on these connections to create spatial invariance. Skip-connections can be represented as a special operation within the broader representation of layers connected in series. Although each of these ANN structures are simply special cases of more generic structures, they empirically yield valuable benefits during the learning process. The brain also shows structural specialization with hierarchical organization in task-specific regions, such as the visual cortex (Rash et al., 2016), suggesting the benefit of structure for guiding learning.

Deeps ANN architectures are usually hand-designed, static during training, and used generally across many related tasks within a domain. These architectures have begun to approach the desired specialization through their hand-design: however, such deep ANNs are known to be over-parameterized (Ba & Caruana, 2014; Frankle & Carbin, 2018), using more parameters than necessary for their performance on a specific task. Over-parameterization seems to help gradient descent of parameters of an ANN converge to the global optima for a given training dataset, task, and objective function (Zou et al., 2020), particularly given that the architectures are reused for many tasks due to prohibitive engineering costs for further specialization. However, this over-parameterization also comes at a cost of space and time efficiency, during training and deployment. Using even more specialized architectures could reduce this cost and increase the efficacy of the ANN in a specific task.

Automated specialized design of ANN architectures is achievable through structural learning. We define structural learning as optimizing both the architecture and the parameters of that architecture in a single process. As shown in this work, most of this form of structural learning used in practice has taken the forms of pruning, or the removal of parameters to reduce network size while maintaining performance, and Neural Architecture Search (NAS), or the automated search of architectures within a search space. Relatively few papers have pursued developmental structural learning, which involves creation of new structures or connections within the network.

In this review, we take a biologically-inspired view of structural learning in ANNs. The most recent known analysis of structural learning in both biological neural networks and ANNs was completed over two decades ago (Quinlan, 1998). Since then, both artificial intelligence and neuroscience research have advanced exponentially, prompting the need for a modern analysis. The links between biological and artificial structural learning have already been well-defined for the modern neural and biological modeling communities, but a disconnection still exists between these communities and deep-learning with ANNs.

In an effort to surpass the structure design bottleneck, we go back to the roots of ANNs in biological nervous systems in order to define a framework for structural learning in Section 1.1. We summarize the biological mechanisms which resemble the patterns of algorithmic components seen in structural learning literature in Section 2. We survey methods of structural learning in ANNs already existing in literature, providing broad statistical overviews in Section 3 as well as detailed tables of features over our selected corpus at the end of the paper, then diving deeper into each operator of our framework across implementations in Sections 4 and 5. We finally give suggestions on how to progress forward on current challenges in Section 6.

Our aim in this work is to shed light on a similar set of research directions which are already well established in disjoint fields but not yet often linked. Using a common language for structural learning may help connect the many subcommunities of ANN research that are already researching similar approaches to structural learning with different implementations, abstractions, and goals.

1.1 Framework Definitions

In order to understand structural learning, we propose a definition which, in this work, will be used to define the scope of study. We consider structural learning to be a change to a neural network through any of the following four atomic operators, referred to subsequently as the **neural operators**:

- **Neuron creation**: the addition of a unit to an ANN,
- **Neuron removal**: the removal of a unit from an ANN,
- **Synaptogenesis**: the creation of a non-zero weight between two units,
- **Synaptic pruning**: the removal, or change to zero, of a non-zero weight between two units.

We purposefully open the definition of neuron creation and removal to different unit types: in many structural learning algorithms, entire groups or "units" of parameters are added and removed together. Units may be as simple as a neuron in an ANN, or be another higher level structure such as a convolutional filter, channel, layer, or module of layers. A unit contains a portion of the parameters within the network and provides basic organization to make the construction of a network more tractable. While the basic computational units in both the brain and in ANNs are conventionally called neurons, we use unit as a more generic term that can include any structure within a network being used as nodes in a graphical representation of the structure.

The dominant form of learning in ANNs is weight training through error backpropagation, which optimizes synaptic weights and neuron biases in an objective function landscape using gradient descent. In this article, we aim to characterize the impact of these neural operators on learning, which is often coupled with weight training in a bilevel optimization process. We observe, for example, that the neural operators change the shape or dimensionality of this landscape. Objective function optimization is already inherently difficult due to nonconvexity: structural learning further adds considerable complexity to the search, because the effects on the landscape can only be locally and acutely evaluated.

The characterization of a neuron in an ANN lies in its layer type, activation function, and connections. In conventional deep learning, the layer type, activation function, and pattern of non-zero connections are kept constant while the parameters controlling the weight of connections are optimized via backpropagation. The neural operators allow for change in the non-parameter characterization of each neuron, including its connectivity.

We briefly discuss the biological structural learning processes that inspired our neural operators in the following section.

2 Development in the Biological Brain

Many structural learning algorithms for ANNs are inspired by biological neural development at the cellular level, from synaptic pruning (LeCun et al., 1990) to neurogenesis (Martin & Pilly, 2019). To better understand the role of and possibilities for structural learning in ANNs, we compare the four previously defined neural operators of neuron creation, neuron removal, synaptogenesis, and synaptic pruning to their biological counterparts. Neural development in biological nervous systems and the brain is a complex process, and a comparison with artificial representations necessarily requires simplification, but even a simplified view of biological structural learning of microcircuitry in the brain offers insight for the artificial process.

We focus on the epigenetic timescale of neural development as it occurs in individual organisms in two distinct stages: early development and structural plasticity in the mature brain. Both have been used as inspiration for structural learning; for example, the self-organizing ANNs in Li et al. (2004) are based on early development while the synapse modulation method proposed in Aljundi et al. (2018) is inspired by learning over the human lifespan. Both periods are relevant for structural learning in ANNs, offering different views on the role of neurogenesis, synaptogenesis, synaptic pruning, and neural cell death.

2.1 Neurogenesis and neural migration

Neurogenesis is abundant and rapid during early development. Starting from around 5 weeks of embryonic development, the human brain will develop almost all of the roughly 86 billion neurons present in the adult brain (Azevedo et al., 2009). After approximately 28 weeks of gestation, this vast neurogenesis slows down considerably, with only a few regions continuing neurogenesis after birth (Malik et al., 2013; Urbán & Guillemot, 2014).

As this massive neurogenesis is taking place, cells also begin to move to specific locations, organizing into distinct regions and specific structures (Wu et al., 1999). Chemical and sensory cues guide this neural migration, which is essential for the functioning of the brain. Neurons which started in distinct germinal zones may travel through specific complex routes to reach their final destination, resulting in different composition of neural types in different sections of the brain. By 18 months of age, much of the structure of the brain will be set by early neurogenesis and neural migration (Ackerman, 1992). Structures like the cortical columns, which inspired convolutional neural networks, result from this neurogenesis and neural migration of early development (Rash et al., 2016).

Neurogenesis becomes much less common once early development is over; indeed, it was previously believed that there were no new neurons added to the adult human brain. There is growing evidence that neurogenesis occurs in the adult brain (Ming & Song, 2011; Ernst et al., 2014), but has only been identified in specific regions. While the mechanisms behind adult neurogenesis are not yet fully understood, nearby neural activity has been identified as a factor, supporting the idea that the brain can create new neurons in response to needs which are not fulfilled by existing neural circuitry (Ghashghaei et al., 2007).

Beyond their movement and connection to other neurons, neurons are also capable of change during their lifetime. Learning can produce nonsynaptic changes of neural membranes, such as membrane conductances which modulate excitability (Mozzachiodi & Byrne, 2010). While artificial neurons already modulate their bias during conventional, non-structural learning, these changes to the neural cell body could be seen as a more drastic change to the neural activation function.

From an artificial perspective, neurogenesis and neural migration can be considered as a single event in modern artificial neural networks, which rarely consider the physical placement of neurons. The neurogenesis of early development is therefore akin to neural architecture search space definition; the fundamental components of learning are in place but not yet properly wired. Adult neurogenesis in the biological brain, on the other hand, takes place in specific sections where existing structures and well-defined activity patterns abound. An artificial analogy of adult neurogenesis could be an expansion or modification of the possible synaptic weight space, such as the addition of a new layer, convolutional filter, or individual neuron.

Early development sets the stage for cognition and learning in the mature brain. Neurogenesis on a massive, rapid scale, with new neurons being created, differentiating, and migrating, is a hallmark of early development and is relatively unique to this period. Synaptogenesis next takes this stage in early development, with axons branching through complex routes to create the immense connectome of the brain and nervous system.

2.2 Axon migration and synaptogenesis

During early development, as new neurons are created and migrate to specific sections, connections between these new neurons begin to establish the neural circuitry of the brain. Shortly after neurons differentiate, genes which encode proteins necessary for synaptogenesis are activated (Waites et al., 2005). The creation of connections between neurons begins during development but continues well into early postnatal life and is thought to contribute significantly to learning and memory in adults (Waites et al., 2005).

After differentiation, neurons project axons with growth cones which also migrate, following paths shaped by combinations of attracting and repelling signaling molecules exposed by surrounding cells (Erskine & Herrera, 2007). For almost all types of neurons, their singular axon will serve as the output line of communication, carrying an electrical discharge to other neurons. Axon development and migration happen throughout early development, with some axon growth cones traversing long and complex routes; the longest axon in humans is the sciatic nerve, reaching over a meter from the base of the spinal column to the foot.

In the central nervous system, innumerable connections form between neurons in intricate patterns during early development. As an axon growth cone extends beyond the neuron, the neuron will also extend many other branches around its cell body: dendrites, which receive electrical stimulation from other neurons. Once an axon growth cone and dendrite branch are sufficiently close, synapses (individual connections) may be created (Yoshihara et al., 2009). The formation of an individual connection, synaptogenesis, is regulated by chemical signals, activity of both the presynaptic and postsynaptic neurons, and genetic factors (Ackerman, 1992).

In early development, as in the mature brain, nearby axons and dendrites will exchange a number of signaling molecules which, in tandem with other cues, make neurons more receptive to synaptogenesis. The fate of newly formed synapses will be decided by the activity of the connected neurons, determining if they will be stabilized or eliminated (Waites et al., 2005). A single neuron will form many similar connections which compete for stability as failed synapses close and the axon or dendrites retract.

While genetic cues inform many of the initial paths of axon migration and branch, neural activity sculpts neuronal arbor growth and synapse formation. Disruptions to neural activity have been shown to drastically change connectome organization, from the level of individual synapse formation to entire axon branch placement (Pfeiffenberger et al., 2005). In developmental synaptogenesis, neural activity is a contributing factor amongst others for synaptogenesis, but as the brain matures, activity becomes an important trigger for further synaptogenesis.

In the mature brain, synaptogenesis plays a central role in associative learning and memory. Pharmaceutical treatments such as antidepressants have been shown to produce significant increases in the number of synapses in the adult brain (Nelson & Alkon, 2015). New neurons in the mature brain may develop connections in a different way than neonatal neurons do, first developing input synapses on their dendritic tree to observe incoming connections before firing action potentials (Kelsch et al., 2010). In this way, the connections formed by new neurons in the mature brain may be informed by existing neural circuitry before adding new connections and functionality.

From an artificial perspective, the synaptogenesis of early development can be considered as similar to developmental neurogenesis; the preliminary neural circuitry is defined largely by genetic cues which create the possible space for future learning through synaptic tuning. While the synaptogenesis of early development can depend on activity, it has also been demonstrated in the absence of activity, pointing to circuitry designs predetermined by genes (Waites et al., 2005). In the mature brain, however, activity is an integral part of the integration of new neurons and synapses into the neural circuitry. New neurons will listen to nearby activity before activating, forming downstream synapses based on observed activity patterns. As with neurogenesis, synaptogenesis sets the stage for learning to take place, but it is with the stabilization and elimination of parts of this neural circuitry where specific learned patterns in the brain are shaped.

2.3 Synaptic pruning

Connectivity patterns in the mature brain are learned through a process of correlation detection, synaptic competition, and eventual synapse elimination, i.e. pruning. The pruning process has been identified as a crucial component of learning and of typical neural functionality. It is widespread in the central nervous system but has also been observed in the peripheral nervous system (Riccomagno & Kolodkin, 2015). Pruning occurs at a massive scale during development but continues throughout lifetime (Faust et al., 2021).

The prevalence of pruning in the developmental and adult brain indicates the presence of numerous redundant or excess connections formed during development. The reasons for these exuberant connections are not fully clear, but there are multiple possibilities, one being physiological or structural roles of these excess connections during early development (Riccomagno & Kolodkin, 2015). More importantly from an artificial perspective, however, is the information available for neural circuit determination. During neurogenesis and synaptogenesis, possible communication between cells is limited to chemical signals and a limited number of connections. Excess connections offer a rich information pipeline for neural circuits to refine into specific functional patterns.

The information pipeline of highly-connected neural circuits offers a way to detect correlation between connected neurons. Experimental evidence has demonstrated that neural activity regulates synapse elimination (Waites et al., 2005; Faust et al., 2021) showing that correlated synaptic activity informs about the relevance of synaptic connections and influences synapse stability and elimination as postulated by Hebbian theory (Hebb, 1949). Neurons with high levels of correlated activity will retain connections while others are pruned, forming specific circuits based on activity patterns.

The development of the visual system demonstrates the importance of activity for eventual circuit formation. Highly correlated patterns of spontaneous activity in the form of retinal waves occur throughout development, driving activity-dependent pruning in the visual system (Katz & Shatz, 1996). Experiments in ocular deprivation have shown that a deprivation of activity in one eye can lead to structural changes in the visual cortex which result in a lack of response to new activity in the covered eye (Hubel et al., 1977). The brain undergoes significant rewiring in response to specific novel activity patterns, in the visual cortex and elsewhere.

Beyond the local information of presynaptic and postsynaptic neural activity, other signaling mechanisms influence synapse elimination. Microglia in particular have been identified as an important mediators which promote and constrain synaptic elimination (Wilton et al., 2019). The microglia can tag unwanted synapses with specific proteins leading to their elimination (Stevens et al., 2007). While the full internal mechanisms of the glial cell tagging decision are not yet completely understood, the molecular mechanisms involved in synapse elimination are spatially and temporally dependent, and disruption to these mechanisms has been linked to brain dysfunction (Riccomagno & Kolodkin, 2015). From an artificial perspective, this motivates the design of pruning decisions which are made not only based on internal factors such as synaptic weight and activity information, but through complex and dynamic policies which account for various information over entire regions of the brain.

Active neurons can have a large number of synapses pruned with little effect on the neural cell body. However, in the event of pruning along an entire axon branch, the branch will retract or be eliminated from the cell (Bishop et al., 2004). Retrograde signaling mechanisms pass information from distant axon heads back to a neuron cell body to inform of pruning events and to initiate an appropriate response (Harrington & Ginty, 2013). This may involve restructuring of the axon or dendritic tree of the neuron, but can also in some cases elicit neural cell death. In this way, sufficient synaptic pruning and appropriate signaling can lead to complete neuron removal, as we examine in the next section.

2.4 Neural cell death

Neural cell death can be distinguished into two main types based on timing: a common early cell death occurring in neural progenitors and immature neurons, and a late cell death which occurs in postmitotic neurons (Vanderhaeghen & Cheng, 2010). There are many mechanisms and underlying causes for neural cell death, with some mechanisms shared between the two types, but in many ways, they are distinct processes. We consider both types, aiming to clarify the decisions leading to cell death in both cases.

Early cell death is a necessary part of neural development and occurs either before full differentiation of the neuron and its projections or during synaptogenesis (Yeo & Gautier, 2004). This process is a mechanism for cell number and quality control, removing unwanted cells and those damaged by proliferation. While this contributes to the final neural circuit through cell number control, experiments on the prevention of neuronal cell death during development have demonstrated that phenotypically similar circuitry can arise despite the presence of excess neurons (Buss et al., 2006). However, neural cell death does regulate mature functional circuitry through controlling competition over factors necessary for cell survival such as chemical signals from glial cells (Yeo & Gautier, 2004). Notably, early neurons require a constant signal to prevent cell death; in this way, elimination can be seen as the default end of new neurons with persistence as the exception.

Early neural cell death is abundant in early development, in tandem with the massive amounts of neurogenesis, and it also accompanies adult neurogenesis. Adult neurogenesis has been observed in the olfactory bulb but from the new neurons generated, only 50% will survive for extended periods of time and be integrated into the existing circuitry. Interesting, in this case, neural activity was demonstrated to be am important factor in determining cell survival (Kelsch et al., 2010).

From an artificial perspective, inspiration from biological early neural cell death can inform the decision of elimination of new or untrained operators such as layers. As the parameters of these operators are not yet informed, evaluating their potential for removal can be difficult yet rewarding. The biological process of creating a large excess of cells and eliminating quickly can serve as motivation for artificial structural learning.

Programmed cell death in mature and connected neurons is a much rarer event in healthy brains. Principle causes of postmitotic neural cell death are the death of other connected neurons (transneuronal degeneration), axon degeneration (Wallerian degeneration), or the result of interaction with components in the extracellular matrix originating from other neurons and glial cells (Fricker et al., 2018). In the adult brain, these events typically occur in response to an injury or as part of a neurogenerative disease. Transneuronal degeneration occurs relatively frequently in young animals where new neurons are still being formed, but is much less common in adults. Wallerian degeneration mostly occurs in response to axon injury, inflammation, or genetic impairment. The signals which can elicit mature cell death are numerous but are most common in neurogenerative diseases such as Alzheimer's disease (Fricker et al., 2018).

As the death of highly-connected, postmitotic neurons is such a rare event in healthy brains, the main role of neural cell death in the learning process occurs early on, during or even before synaptogenesis. This indicates that, as opposed to individual synapses which are often pruned throughout learning, mature neurons are costly to remove. While this principle correlates well to highly parameterized ANN operators, it is much less costly to remove an artificial neuron than a real one, as the costs of cell removal and replacement neurogenesis are greatly reduced.

2.5 From Biological to Artificial

It is a drastic simplification to summarize these complex processes in the brain, which are still not fully understood, with the four previously defined neural operators. However, artificial neural networks have different information and energy requirements than biological neurons. Artificial neurons which are represented as distant can be connected just as easily as neurons which are represented as adjacent. Information is transferred between artificial neurons without loss; synapses don't have to be protected from possible interference. There is no bandwidth limitation for individual synapses nor additional information gained from synapse location, so the relationship between presynaptic and postsynaptic neurons can be summarized by a singular continuously-valued synaptic efficacy measure. The similarities and differences between biological neurons and the artificial models inspired by them are constantly evolving and being better understood, but it is clear that their fundamental differences require different learning operator definitions.

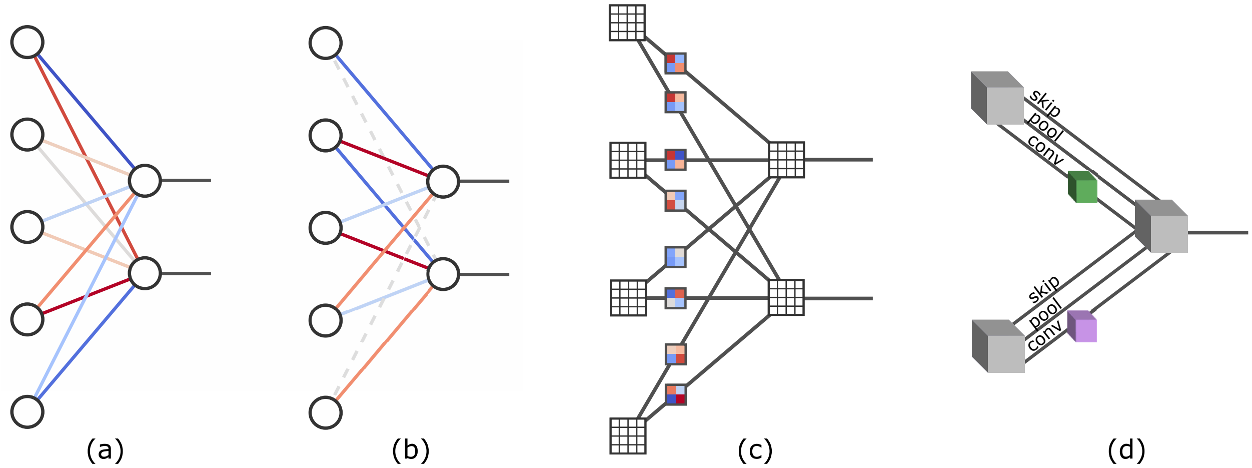
The result of these simplifications is that many decisions made by cells in the biological brain are aggregated into the neural operators. The variety of information sources for cellular decisions are streamlined in ANN operators; chemical signaling is mostly ignored in ANNs as is cellular contact. However, recent trends in ANN literature have begun to explicitly reintroduce the sort of information pathways used in the biological brain, such as the role of non-neuronal cells (Azevedo et al., 2009), Hebbian synaptogenesis (Dai et al., 2019b), and activity-dependent neuron removal (Kang & Han, 2020). Understanding the complexity of these structural changes in the biological brain can bring new perspectives on the formulation of neural operators in ANNs.

3 Characterizing Structural Learning in ANNs

We review relevant implementations of these biologically-inspired operations in existing ANN literature, focusing on novel algorithms for changing the structure during the course of training an ANN. Using the definitions of the neural operators of neural creation, synaptogenesis, neural pruning, and synaptic pruning allows for characterization of these algorithms from a structural learning perspective, drawing links between different implementations of the neural operators. Some instances of these neural operators are brain-inspired by design, while others are incidentally correlated to biological operations.

In general, ANNs are constructed as a composition of individual components, layers, in a directed and usually acyclic graph. We refer readers to Goodfellow et al. (2016) for a full review of the different basic components of modern ANNs, such as convolutional, pooling, and normalization layers.

The abstraction from individual biological neurons to artificial feed-forward neurons, the most basic type, is clear. However, convolutional neural networks (CNNs) have two levels of abstraction, depicted in Figure 1. The first is by considering a CNN as a special case of a feed-forward neural network (FFNN) with non-local weights zeroed out and local weights shared. The other paradigm moves the neuronal unit up to a higher level, considering a channel in each convolutional layer to be the neuron abstraction. Connections between channels represent synapses and are parameterized by a small matrix kernel rather than a single value, and the activation within the neuron is a larger matrix rather than a single value. While the lower level paradigm continues to multiply the singular activation coming as input to each synapse by the single shared synaptic weight value, the higher level paradigm performs a convolution of the kernel parameter matrix over the incoming activation matrix before summation across connections. Using the lower level paradigm of Figure 1b for structural learning translates to partial-area convolution or changing kernel characteristics such as size and striding across neurons and is denoted as "Kernel shape" in Table 1. Using the higher level paradigm of Figure 1c becomes adding and removing channels as neurogenesis and neural pruning, respectively, denoted as "Filter/channel" in Table 1. An even higher level paradigm is super-networks in NAS, shown in Figure 1d,



(a) (b) (c) (d)

Figure 1: Levels of abstractions of found in structural learning algorithms. From the level of **(a)** feed-forward neurons, there are the two levels of convolutional paradigms: **(b)** on the same level as FFNNs with weight-sharing and systematically pruned connections, and **(c)** on a higher level with a parameter matrix on each synapse and an activation matrix within each neuron, compared to singular values respectively on the lower level. The shared weights in **(b)** are referred to as a kernel, which are the small matrices parameterizing each edge in the higher level paradigm in **(c)**. On an even higher level is the NAS super-network paradigm **(d)**, where each edge may contain parameters based on the operation type, like the parameter-less skip-connection and pooling and the parameterized convolution operations shown. Higher levels have higher dimensionality, structure, and complexity within each node and edge.

denoted as "Layer(s)" in Table 1. We consider works on structural learning which function on these different levels of abstraction.

3.1 Scope of Study

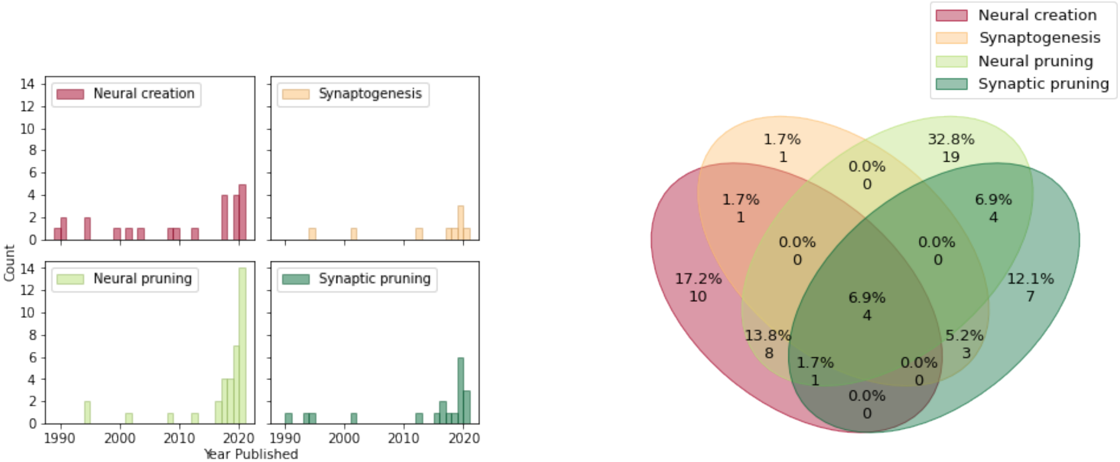
Our 59 collected papers from ANN literature each demonstrate at least one of the four neural operators, neural creation, synaptogenesis, neural pruning, and synaptic pruning, over the course of training an individual ANN via an optimization method. We focus on standard ANNs with parameterized connections, generally with feed-forward, convolutional, or recurrent layers.

ANNs simplify the biological complexity of connection strength between two neurons into a single continuous value. We consider all changes to this weight, which could represent biological structural change, to be weight training, not structural learning. This includes soft structural methods that do not make any discrete changes to the architecture and thus no changes to the effective dimensionality of the objective function, such as Rusu et al. (2016). Only hard structural methods are evaluated, including those which involve a discretization step after soft structural learning, such as Liu et al. (2018).

While synaptogenesis necessarily follows neurogenesis in the brain through different processes, we consider the non-learned creation of connections of a new unit to be encapsulated within neuron creation in ANNs, such as adding a neuron with default connections to all neurons in the previous and subsequent layers. Thus, in our characterization, algorithms are only labeled with both neuron creation and synaptogenesis if there is a connection learning component for existing units in addition to a process for creating new units with a default set of connections.

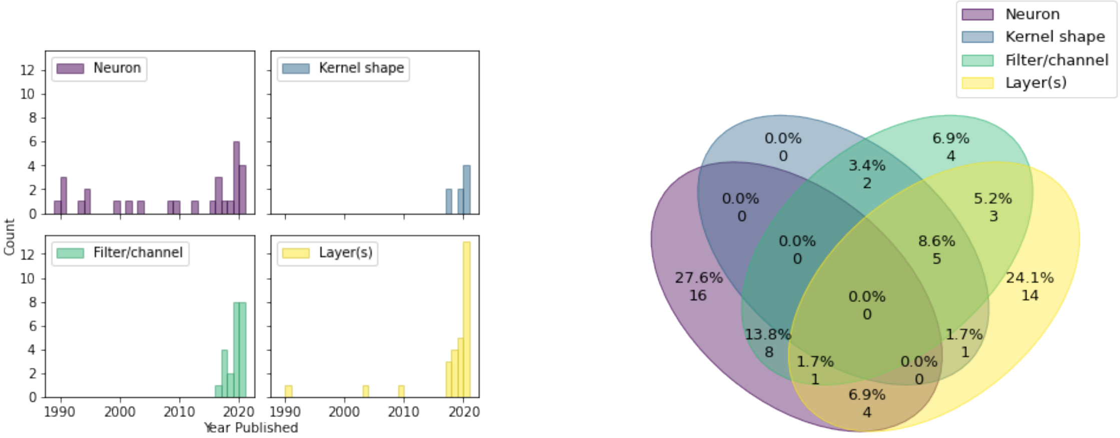
Many structural search papers enumerate all possible options throughout the learning process in order to select the best combination of structures to build the final architecture. For example, NAS algorithms initialize all possible layers with all possible connections and learn which ones are best. We consider this as synaptic and/or neural pruning, rather than neuron creation and/or synaptogenesis.

A biologically inspired family of algorithms for discovering ANN structures automatically are evolutionary algorithms. These algorithms model biological evolution by maintaining a population and recombining the genes of successful candidates to produce more candidates. Some such methods meet our criteria for



(a) Occurrence of operators over time.

(b) Co-occurrence of operators within the collected corpus.



(c) Occurrence of unit type over time.

(d) Co-occurrence of unit type within the collected corpus.

Figure 2: Characterization of neural operators and unit type used in our selected corpus of papers.

structural learning by persisting network weights across individuals within the population (Elsken et al., 2017; Ci et al., 2021). However, many evolutionary methods maintain a population of individuals with different structures and weights, such as Lu et al. (2020) and Stanley & Miikkulainen (2002), and do not study individual structural learning operators on common weights. Similarly, other search methods that do not persist weights during structural changes are not included. This limits the collected corpus to one-shot algorithms, which pursue a more continuous path in the changing objective function space during the co-optimization of architecture and parameters.

Finally, we exclude architectures with self-gating mechanisms, such as LSTMs (Hochreiter & Schmidhuber, 1997) and transformers (Vaswani et al., 2017). These neural networks change their own structure, but in a transient and input-dependent manner, occurring on a shorter timescale than structural learning. This is more akin to neural circuitry gating and modulation Lindsay (2020).

3.2 Characterization of the Collected Corpus

The goal of this work is to characterize structural learning, rather than document all uses of any of the four neural operators in the vast ANN literature. Our selection of 59 seminal works is therefore not comprehensive and is rather intended to be a qualitatively representative sample, as there are orders of magnitude more papers in the subfields of NAS and pruning than in the developmental structural learning space. Completing a more rigorous collection process for complete quantitative characterization would be prohibitively nuanced, for reasons including inconsistent vocabularies surrounding structural learning used across subfields and subtle differences between novel structural algorithms versus applications of existing techniques to new architecture types and task domains, and thus is beyond our scope. The 59 works in the collected corpus provide a diverse and informed view of the structural learning literature.

We refer the reader to other reviews for a more complete view of specific domains. Deng et al. (2020) covers pruning and many other methods for model compression. Hoefler et al. (2021) discusses sparsity in neural networks, which is usually achieved via pruning. Elsken et al. (2019) describes the state of NAS as a burgeoning sub-field, while Xie et al. (2021) is more recent and focuses on weight-sharing NAS. He et al. (2021) covers AutoML, which encompasses methods that contribute to automating the machine learning pipeline beyond parameter tuning and thus includes structural learning as well as data preparation, hyperparameter optimization, and other types of architecture search. Stanley et al. (2019) gives an overview of the use of evolution for neural network optimization, including methods which change neural structure over generations. Evolutionary developmental methods which evolve rules for structural learning are reviewed in Miller (2022). Parisi et al. (2019) discusses methods for continual learning in ANNs, many of which use forms of structural learning to increase the information capacity of networks performing multiple tasks. Our goal in this article is to offer a coherent synthesis of works from this different domains under the framework of structural learning.

The general statistics for neural operators and units across the collected corpus are shown in Figure 2. These plots show trends and frequencies of characteristics across the collected corpus over time, although not necessarily representative of structural learning without the aforementioned biases in our sample collection. Neural creation and synaptic pruning appeared earlier, while synaptogenesis and neural pruning have become more popular in recent years, shown in Figure 2a. Most papers perform a single operator, but a significant portion perform a combination, shown in Figure 2b. Operating on units of standard neurons has been consistently used over time, while the arrival of deep learning architectures is evident in Figure 2c with structural learning over layers and convolutional units of kernel shape and channels becoming more popular after 2010. The impact of deep learning is also evident in Figure 2, with a significant increase in pruning methods while the automatic creation of new units or connections did not undergo a similar expansion of interest. We expect this is due to the complexity of the creation decision, which we explore in section 5. Finally, as with neural operators, operating on a single unit type is common, but more algorithms are permitting structural learning across different units and layer types, shown in Figure 2d.

Further trends over the qualities of each algorithm in the collected corpus sorted by publication year are detailed in Tables 1-2 at the end of the paper. The earliest algorithms were designed before convolutions and recurrent layers, but now most algorithms are designed for and demonstrated on ANNs with convolutional layers, with some additionally applied to dense layers or recurrent layers. Regarding tasks, nearly all algorithms demonstrate on supervised learning tasks, particularly image classification in more recent years. Some specific techniques are applied to multi-task, continual learning, more specialized computer vision, natural language processing, sequential data, and reinforcement learning tasks: all of these pose additional challenges above the relatively simple image classification or tabular dataset classification baselines. We discuss existing structural learning methods in the context of each operator in Sections 4-5 and relevant implementation trends in Section 6.1.

Each of the four neural operators have different considerations and effects within structural learning. Performing synaptic pruning or neuron removal on an ANN is more tractable than their additive counterparts, because their structural learning search space is naturally defined by the existing network structure rather than being open-ended. As the two pruning operators have also been more commonly used in literature, we will consider them first in Section 4 before discussing the two creation operators in Section 5.

4 Synaptic Pruning and Neural Pruning

Our main questions to characterize pruning in our selected papers are:

- **Goals of Pruning:** Why remove connections or units from an ANN?

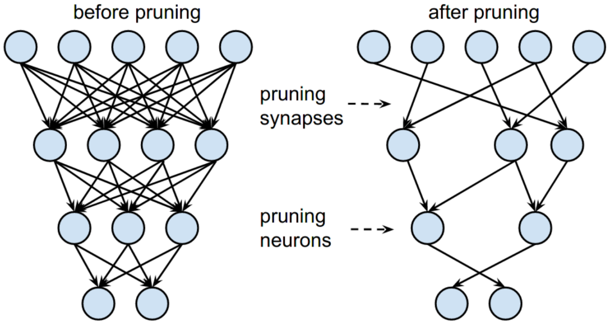


Figure 3: Neural pruning and synaptic pruning in a fully connected ANN (figure reproduced with permission from Han et al. (2015)). When a neuron is pruned, as in structured pruning, all incoming and outgoing synapses are also removed. Synaptic pruning additionally removes synapses between unpruned neurons and is also referred to as unstructured pruning.

- **Architecture Specificity:** How has pruning been adapted to different ANN architectures, such as dense layers and convolutional layers?
- **Measuring Entities:** How is the search performed? How is importance of a connection or unit measured in order to make the pruning selection?
- **Scheduling Pruning:** When and how often does pruning occur with respect to optimization of the unpruned parameters?
- **Handling Pruned Entities:** How should pruned connections and units be handled?
- **Impact on Weight Optimization:** What is the impact of pruning on optimization of parameter weights?

The following subsections consider the implementations of synaptic pruning and neural pruning found within our corpus. We will first discuss a subset of these questions for pruning in general in Section 4.1 and then dive further into specifics of all questions for synaptic pruning in Section 4.2 then neural pruning in Section 4.3. Finally, we will draw conclusions and connect our discussion back to biology in Section 4.4.

4.1 Pruning Generalities

Pruning aims to remove parameters in an optimal manner, but their effect is not independent, and evaluating the effect of all possible combinations of parameters to prune on the network performance is not computationally feasible. Not only do the combinations grow exponentially with the size of the network, but the cost of completely evaluating each one by locating the new optimum through optimization to convergence is prohibitively expensive. Thus, algorithms generally make an approximation by estimating the effect of removing each parameter independently from the local landscape of the objective function on training data (Hoefler et al., 2021). This naturally leads to greedy pruning decisions from the static set of weights, so parameters that currently seem beneficial to remove are selected at the time of pruning. Algorithms can only assume that this process approximates the globally optimal solution for what to remove from a network. Many Neural Architecture Search (NAS) papers formalize the difficulty of optimizing architectures and network weights simultaneously as a bilevel optimization problem, to be discussed further in Section 4.3.

Goals of Pruning: Pruning an ANN, or the removal or zeroing of portions of the network, to make it more efficient in space or time while maintaining performance is one of the many orthogonal ways to perform network compression. Using smaller networks with the same performance is particularly desirable for real-time applications and resource-limited devices such as mobile. Pruning large, over-parametrized networks to find their best-performing sub-networks is also a method for finding a high-performing architecture for a given network size. These naturally lead to applications of pruning in multi-task learning, where portions of the network are shared between tasks and others are task-specific, and NAS, where the architecture is optimized on a higher level.

Measuring Entities: Each entity must be evaluated in order to make an informed pruning selection. The final effect of changing an entity at a given training step cannot be predicted. Entities to prune may be selected by a salience metric computed per-entity designed to estimate the effect of removing the entity on the performance. Masks can also be used, introducing a new parameter for each entity that controls whether it is active of not. This is particularly useful for ephemeral pruning, where pruned entities may be reactivated later in the course of learning. The mask may be discretely controlled by a salience metric or relaxed to continuous values that can be trained via optimization. Finally, regularization can be used to encourage sparsity of network or mask parameters during gradient descent steps by adding a magnitude-penalizing term to the objective function.

Once the selection metric or mask is evaluated, the algorithm may then use either ranking or thresholding, at either a local or global scale, to discretize the pruning decision for each parameter. Ranking allows a defined proportion to be pruned, which may be particularly desirable for hard time or space constraints of the final ANN, while thresholding allows for only parameters meeting a score criteria to be pruned, which can be more consistent over iterative pruning phases. Local pruning makes the ranking or thresholding decision within a structure such as a layer, which makes comparisons more homogeneous, while global pruning makes these comparisons over the entire architecture, which allows the algorithm to determine the pruning level for each structure automatically.

Scheduling Pruning: The schedule of pruning determines how it is interwoven with weight updates, including when and how often. Iterative pruning, rather than all-at-once, has generally been more effective (Liu et al., 2021b). It allows the network to evaluate entities in a more intermediate state of pruning, lessening the effects of the assumptions of pruning independence between parameters, high order terms ignored in most metrics, and local objective function evaluations. Most pruning algorithms begin with a large, dense network, while some perform dynamic sparse training, where the initial network is sparsely initialized (Bellec et al., 2018; Wortsman et al., 2019; Liu et al., 2021b).

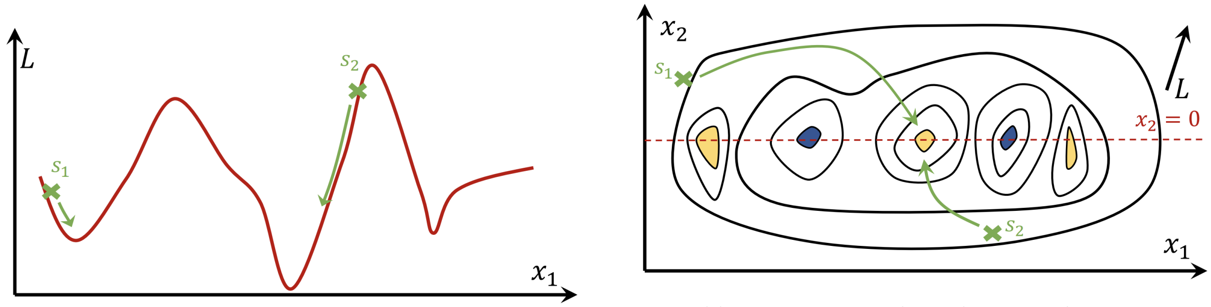
Handling Pruned Entities: After pruning, the pruned entities may be either permanently disregarded or allowed to be revived, which we name ephemeral pruning. We distinguish ephemeral pruning from entity creation if the revival is done with the same method as the pruning and thus is bounded to the same search space as pruning.

Impact on Weight Optimization: Because the objective of pruning is often to at least maintain performance, this is synonymous to reducing the dimensions of the objective function space as much as possible while maintaining or decreasing the cost of the found minimum upon convergence of the objective function. Convergence to a globally optimal point is not guaranteed for standard gradient descent techniques in structurally static networks, but the local minimum found usually has a low enough cost in practice (Goodfellow et al., 2016). Because pruning potentially changes the cost and location of each optimum, this is necessarily an even more difficult optimization than training a structurally static network. Techniques mentioned so far such as iterative pruning and regularization intuitively help ameliorate negative effects of the discrete changes in the shape of the objective function while performing gradient descent in parallel to pruning, thus aiding in the search (Hoefler et al., 2021).

Pruning may be categorized by the level of structure used in the pruning process, as shown in Figure 3. Unstructured pruning, which we discuss further in Section 4.2, is the zeroing of any parameters within an ANN and thus is generally considered synaptic pruning for our neural operatordefinitions. Structured pruning, which we discuss further in Section 4.3, is the removal of entire units and their associated parameters within a network, such as channels, neurons, or layers, so it is synonymous with neural pruning.

4.2 Pruning Connections

Goals of Pruning: Synaptic pruning is the lowest level and simplest form of pruning; it intends to remove parameters, which each represent a connection, without any pattern or structure. The earliest pruning papers began with unstructured pruning (LeCun et al., 1990; Hassibi et al., 1993). Pruning synapses in a



(a) Single parameter gradient descent, where continuing descent from s_1 results in a local minimum.

(b) Double parameter gradient descent, where continuing descent from either s_1 and s_2 results in descending to the global optimum.

Figure 4: Comparison of gradient descent in a toy loss landscape without and with an extra dimension (figure reproduced with permission from Hoefler et al. (2021)). Transitioning from **(a)** to **(b)** represents creation, while transitioning from **(b)** to **(a)** represents pruning. s_1 and s_2 are potential parameter value(s) at the time of structural change.

network is accomplished by zeroing connections and may allow a lower space complexity of storage by storing the remaining parameters in sparse matrices. However it does not yield significant improvements in time complexity of training or inference without specialized software for accelerating sparse matrix multiplications. However, ANNs after unstructured pruning can outperform dense models of the same memory footprint (Zhu & Gupta, 2017), showing that the pruned models can be more specialized and effective, The main applications of synaptic pruning are for network compression (LeCun et al., 1990; Hassibi et al., 1993; Han et al., 2015; Guo et al., 2016; Lee et al., 2019) and for balancing synaptogenesis (Puma-Villanueva et al., 2012; Bellec et al., 2018; Wortsman et al., 2019; Dai et al., 2019b;a; Du et al., 2019). It also benefits multitask learning such as in Peng et al. (2021), where only important connections are saved for the current task and the pruned connections are reused for future tasks.

Architecture Specificity: Unstructured pruning is simple to apply to different neuron and layer types, since it can be agnostic to the layer type and just removes single parameters at a time. For convolutional layers, synaptic pruning can mean removing single values within a 2-D kernel (Han et al., 2015; Guo et al., 2016; Bellec et al., 2018; Dai et al., 2019b;a; Du et al., 2019; Lee et al., 2019) or the entire kernel itself (Wortsman et al., 2019). Although each kernel contains more than a single parameter and is thus a higher structure, it can abstractly represent a connection between channels in convolutional layers, similar to a connection between neurons in dense layers (Wortsman et al., 2019), and cannot be explicitly removed from the 4-D convolutional matrix, only zeroed, so it is more akin to unstructured pruning.

Measuring Entities: Salience metrics try to estimate the importance of each connection in order to prune unimportant connections. They can be as simple as the absolute parameter value itself such that weights near zero are pruned (Han et al., 2015; Dai et al., 2019b;a), which is intuitive as they would have a lower impact on the ANN than weights with stronger magnitudes. More complex calculations include gradients, which measure sensitivity of the network outputs with respect to each parameter, so low values signify unimportance. Any metrics requiring a derivative of the objective function must be computed with respect to training data. For most modern datasets, computing them for the entire dataset is not feasible, so mini-batches are often used. A local estimate of the effect of removing a single parameter on the objective function may be derived using a Taylor expansion (Laurent et al., 2020). The earliest pruning methods often used up to the second order term, including a full Hessian matrix, within their saliency calculations (LeCun et al., 1990; Hassibi et al., 1993), while current methods usually use cheaper but less accurate approximations with at most first-order derivatives over a single mini-batch (Du et al., 2019; Lee et al., 2019; Peng et al., 2021). For example, the diagonal Fisher information matrix is a first-order metric, scaling linearly with dimensions in computational complexity, that is often used as an approximation of the Hessian matrix, a second-order metric (Peng et al., 2021). The choice of salience metric is usually a trade-off between efficiency and effectiveness, but exactly how useful the precision of saliency is may be confounded with the greedy, iterative pruning process often used.

Masks are also utilized in synaptic pruning. The mask may either be relaxed to be continuous so it can be trained along with the other network weights through optimization (Guo et al., 2016; Bellec et al., 2018; Wortsman et al., 2019; Yan et al., 2019) or remain discretely binary and be controlled with metrics (Dai et al., 2019b;a).

Scheduling Pruning: Regarding the training portion of the schedule, pre-training the large unpruned network is a common first step to a pruning, but some synaptic pruning algorithms avoid this for training time or space efficiency. Lee et al. (2019) forgoes the pre-training step and instead prunes a newly-initialized model using a first-order derivative metric of connection sensitivity. Hassibi et al. (1993) determines the optimal weight update for the remaining parameters after each pruning, but the inverse Hessian required is too expensive to compute exactly for modern ANNs, although estimations can be used (Hoefler et al., 2021). Most neuron-level synaptic pruning algorithms use fine-tuning optimization steps after pruning to arrive at the final architecture with optimized weights.

Handling Pruned Entities: Most metric-based synaptic pruning papers do not allow discrete synapse revival, as computing metrics for inactive parameters often is not possible. However, masked selection techniques are more amenable to ephemeral pruning of connections, which may help ameliorate the negative effects of greediness. Guo et al. (2016); Bellec et al. (2018); Dai et al. (2019a); Wortsman et al. (2019) use dynamic connection search, allowing previously pruned connections to be revived if they show a resurgence in parameter importance later in training.

Impact on Weight Optimization: Synaptic pruning may decrease the dimensionality of the objective function search as finely as a single dimension at a time. The proceeding effect conjectured by Hoefler et al. (2021) is demonstrated in Figure 4. This artificial example supports performing more gradient descent in the higher dimensional space and pruning later in the course of training, at least for traditional pruning occurring intermittently during training. If the network in Figure 4b is initialized to s_1, pruning x_2 before the value of x_1 surpasses the coordinate of the local maximum may lead to convergence to a non-optimal local minimum, but pruning after this point should allow convergence to the global minimum. Pruning non-zero parameters results in a displacement in the search space based on how large in magnitude the pruned parameters were. Magnitude-based pruning has a minimal but not negligible effect on the displacement of the location in the search space, and may also be more greedy. The performance change due to this displacement does not have clear consequences: the final performance cannot be tractably predicted for any given structural operation. Bartoldson et al. (2020) found that the immediate performance drop after an iterative pruning step was positively correlated with improved generalization and performance of the final network, while Laurent et al. (2020) had conflicting findings, showing loosely inverse correlation. While the objective function is an intuitive tool for optimizing the pruning selection, the interacting effects of the many local assumptions and approximations as well as hyperparameters such as algorithmic details confound its use.

4.3 Pruning Units

Goals of Pruning: Neural pruning is the removal of units within an ANN, such as neurons in fully-connected layers, filters or channels in convolutional layers, or entire layers themselves. Thus, neural pruning has different selection and scheduling considerations compared to synaptic pruning. Structured pruning easily yields both time and space efficiency benefits: entire portions of the network can be removed, thus skipping their computational steps and storage requirements.

Beyond network compression, structured pruning is also a very common abstraction for performing Neural Architecture Search (NAS), an emerging sub-field pursuing the automation of architecture engineering at the layer level. ANN architectures are selected from a predefined search space of possible architectures and evaluated according to a performance estimation strategy. This performance estimation then informs the search, which can be based on evolutionary algorithms, reinforcement learning, gradient descent, or other search methods. Structural learning can be used for NAS by using any of the four neural operators as search operators on persistent networks. Some existing NAS methods function as structural learning, integrating

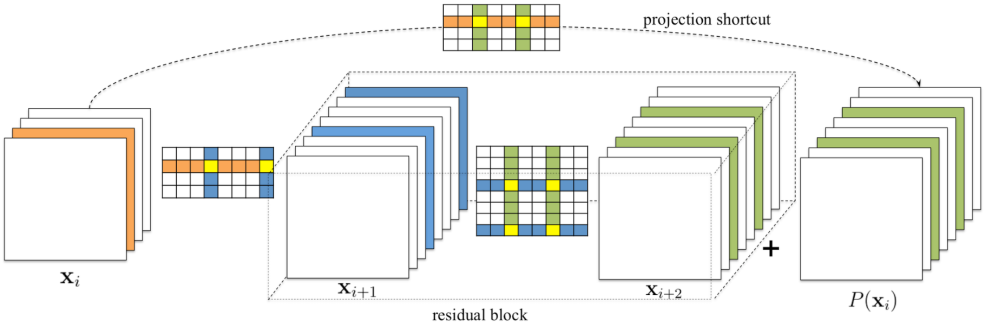
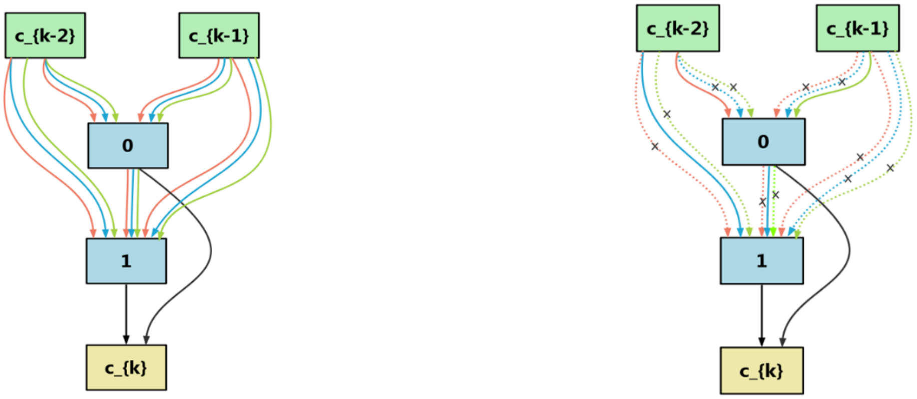


Figure 5: Structured pruning in convolutional ANNs (figure reproduced with permission from Li et al. (2017)). Each stack represents the channels of the activation state, while the matrices represent the kernels, which themselves are *nxn* matrices of parameters, for each convolutional layer, with each row being the kernels for each channel in the previous activation and each column being the kernels for each channel in the next activation. Decreasing the incoming channel count in the first convolution is represented in orange. The blue pruning decreases the outgoing channel count of the first convolution and the incoming channel count of the second convolution. This is synonymous with pruning both the incoming and outgoing connections of a neuron in a dense layer. The projection shortcut performs a 1x1 convolution and is pruned accordingly as necessary for dimensionality agreement.

weight tuning with architectural change to arrive at a final static network in terms of both parameters and architecture. Layer-searching NAS algorithms that use structural learning generally employ a hand-designed super-network that contains all possible layer types over all possible interconnections, then perform the operator of neural pruning after optimizing the super-network to derive the final architecture.

Architecture Specificity: The structural types present within a network require some consideration for structured pruning, depending on the desired level of structure to prune. Many convolutional structural learning algorithms use the higher level paradigm, shown in Figure 1**(c)**, for pruning (Siegel et al., 2016; Li et al., 2017; Du et al., 2019; Wu et al., 2020), as depicted in Figure 5. This allows an algorithm to be applied to convolutional and dense layers, even within the same architecture. Some sub-unit pruning in convolutional layers can still be effective for computational efficiency in contrast to unstructured pruning, such as pruning via striding within kernels (Anwar et al., 2017) and limiting the area of convolution (Dai et al., 2019b). Beyond each layer's type, the surrounding structure and functionality may also be utilized in more specialized pruning: Kang & Han (2020) prunes channels that tend to have low signal after the standard subsequent batchnorm and ReLU operations, while Gordon et al. (2018) regularizes and prunes channels via the batchnorm scaling parameters. Exploiting such known structures in the limited architecture space of those including these structures has a trade-off of search space flexibility and effectiveness within those search spaces.

Measuring Entities: Structured pruning with heuristic search requires metrics or selection methods that can be measured or enforced at the level of desired unit of pruning. Similarly to unstructured pruning, early methods used perturbation-based selection (Narasimha et al., 2008; Puma-Villanueva et al., 2012), where the algorithm measures the performance change from removing each neuron individually and retraining the rest of the network, but this does not scale well with the size of the network. One recent exception used a greedy forward search, where all neurons in a layer are pruned and then useful units are individually added back in Ye et al. (2020). For metric-based pruning of units, a norm across parameters within a unit can be used (Li et al., 2017; Du et al., 2019; Wu et al., 2020) or group-sparse regularizers can tie parameters within a unit and encourage sparsity at the unit level (Pan et al., 2016; Gordon et al., 2018). Selection methods that are only possible on the unit-level include relative metrics of redundancy, where units with redundant functionality compared to others are marked for removal (Siegel et al., 2016). One issue with this approach is time complexity, since it requires a pairwise comparison of all units within each layer. Unit-wide masks, where the mask controls whether each unit is active or not, are also common, such as in Chen et al. (2019); Wan et al. (2020); Guo et al. (2021) at the convolutional neuron level and in most NAS works at the layer level.



(a) Continuous NAS. **(b)** Path-sampling NAS.

Figure 6: Example of a repeated cell of a super-network in NAS Yao et al. (2020). Each colored edge represents a different layer type, such as a skip-connection or convolutional layer. This cell, c_k, receives the output of the two previous cells, c_{k-2} and c_{k-1}. During network training, in continuous NAS, all layer options are used at all connections, and a weighted sum is computed at each node using the architecture parameters. In path-sampling NAS, a path representing a valid architecture is selected within the super-network, which is a form of ephemeral discretization. A common architecture constraint for cell-based NAS is at most one layer option per connection and exactly two inputs to each node. For both types of NAS, the final goal is to optimally prune the network into a discrete architecture within this constraint like **(b)**.

The two main mask-based developmental NAS approaches at the layer level are continuous NAS and path-sampling NAS, as shown in Figure 6. These are used in order to both train the network parameters and evaluate different architecture possibilities, which may be represented as architecture parameters.

In continuous NAS, a continuously structured super-network is trained and then discretized into the desired architecture, as shown in Figure 6a (Liu et al., 2018; Laube & Zell, 2019; Li et al., 2019; Mei et al., 2019; Yan et al., 2019; Bi et al., 2020; Noy et al., 2020; Wang et al., 2021; Roberts et al., 2021). The structural learning, namely neural pruning of the units of layers, occurs at discretization, which may happen once at the end of the search phase or progressively throughout. Before the super-network is fully discretized, it evaluates mixtures of potential layers over potential connections as weighted sums at each activation state, and uses back-propagation of the loss error to strengthen or weaken the architectural weight of each layer. Selecting the architecture from the super-network by the end of the search using learned architecture weights often looks very similar to magnitude-based pruning, selecting the strongest options such that a valid architecture is formed.

For path-sampling NAS, potential paths are discretely sampled from the super-network for training the network's parameters within each path, as shown in Figure 6b (Cai et al., 2019; Veniat & Denoyer, 2018; Ci et al., 2021; Guo et al., 2020; Yao et al., 2020; Guo et al., 2021). During the search process, a discrete path is selected for the training step of each mini-batch, and only the network weights and architecture parameters along that path are updated. Thus, each path is equivalent to performing ephemeral neural pruning for a single forward and backward pass, while persisting and updating weights of the super-network. Path selection may use the architectural parameters as sampling or state-transition probabilities within the super-network (Cai et al., 2019; Veniat & Denoyer, 2018; Guo et al., 2020; 2021) or use a uniform distribution (Ci et al., 2021). At the end of the search process, the highest probability path is selected as the final architecture. While path-sampling NAS allows training passes to occur in a discretized network more structurally similar to the final desired architecture than the continuous mixtures in a super-network, only the parameters on the sampled path, rather than all parameters, have gradient information and thus can be updated per mini-batch.

Scheduling Pruning: As with synaptic pruning, neural pruning has been trending towards iterative methods. Even in NAS, the earliest continuous NAS methods had a single discretization where the pruning neural operatoroccurred, after searching the continuous super-network and before evaluating the discretized architecture (Liu et al., 2018). However, this can lead to a discretization gap, where the shallower continuous super-network and deeper discretized architecture are too structurally different and have uncorrelated performance, due to the continuous parameters not being close enough to the discretized parameters Xie et al. (2021). More NAS algorithms are incorporating progressive discretization (Bi et al., 2020; Wang et al., 2021), among other techniques (Xie et al., 2021), to avoid this gap.

As for the training portion of scheduling, NAS methods also tend to do a full reinitialization and retraining of weights after discretization, whereas most other structural learning methods only finetune the network after the search and pruning process, as noted in Table 2. As for network compression, Liu et al. (2019b) finds that complete reinitialization and training is generally superior while Ye et al. (2020) finds fine-tuning inherited weights is better for pruning algorithms that can effectively use these weights.

Avoiding the pretraining step is more difficult in structured pruning than unstructured, since structural pruning selection techniques often assume all units are trained enough in order to differentiate their function from each other and fairly compare their utility. Pretraining is particularly important for comparing parameterized units with unparameterized units in NAS, like comparing a convolutional layer with a skip-connection at the same location within an architecture. Some relatively zero-cost NAS metrics that measure entire architectures without training have been shown to be effective in identifying good candidate architectures (Abdelfattah et al., 2020), but so far have not been developed on the unit-level for finer structural learning.

Handling Pruned Entities: One classic and simple example of ephemeral pruning on the unit level is Dropout (Srivastava et al., 2014), which is a technique of randomly and independently selecting units to momentarily prune for each batch during training, preventing co-adaptation and overfitting. Structural pruning algorithms use ephemeral pruning in a more principled and informed manner, as in path-sampling NAS.

Impact on Weight Optimization: Structured pruning is less intuitive to consider in the objective function search space, since dimensions are tied through the units containing their corresponding parameters. Thus, multiple dimensions are removed at once for each unit removed. Pruning metrics often gain information through continued training, but this must be balanced with the cost of training units that will be pruned and thus not used in the final network or in further pruning decisions.

Many NAS works have explicitly stated the architecture search as a bi-level optimization problem, where the architecture parameters α are optimized given that the weight parameters w are optimal for any given architecture and constrained to certain sparsity rules S that only permit valid architectures (Maile et al., 2021):

$$\min_{\alpha} \; L_{val}(w^*(\alpha), \alpha) \tag{1a}$$

$$\textbf{s.t.} \;\; w^*(\alpha) = \operatorname*{argmin}_{w} L_{train}(w, \alpha), \;\; \alpha \in S. \tag{1b}$$

This problem is intractable to solve directly, so the process is usually approximated by continuous relaxation of α, persisting weights across the architecture optimization search, and alternating weight updates with architecture updates. The gradient updates to the architectures variables may only be locally approximated: using a second-order estimate of the gradient of α that incorporates an inner update of w^* only gives minor improvements in performance for a higher computational cost compared to first order estimates that approximates w^* as the current w (Liu et al., 2018). See Liu et al. (2021a) for a further discussion on bi-level optimization in ANNs.

4.4 Pruning Conclusions

Even in the brain, more information is available for existing neurons and synapses to be pruned versus creating new units. This trend is noted in artificial algorithms, both in the popularity of pruning versus creation as well as more specifically using masking to propagate gradient information to inactive units, thus turning a structural learning problem into a pruning problem.

Pruning in both the brain and ANNs can improve performance relative to cost: the main costs in both medias are time and space, but the realization of these costs change how efficiency is achieved. For example, an important difference between brains and ANNs is the locality of physical biological signals versus the globality of ANN addressing in memory. Biological neurons are limited by physically transported molecular signals, but do not have a scaling time cost for enumerating the local options. ANNs algorithms have no distance-based cost and thus may make decisions globally, but do have a linearly scaling time cost for each entity that is evaluated.

Pruning itself comes with costs. In the brain, neural cell removal is costly, serving as a possible explanation of the relatively low occurrence of late cell death. The artificial costs are more towards wasted training time of the associated parameters. Pruning algorithms must thus strike a balance of information for pruning and later utility in the network if not pruning versus the cost of obtaining this information.

Pruning has been used throughout the history of deep ANNs to partially automate the structural design. However, it relies upon a predefined and instantiated search space. Currently, most methods explicitly hand-design this search space as the initial architecture. Others, as well as the brain, use creation to dynamically create the search space. These creation methods and their synergies with pruning will be discussed in the following section.

5 Neuron creation and synaptogenesis

Adding to a search space is naturally more difficult than removing, because an indefinite number of ways to add new elements exists, but removal is limited to only existing elements. For the unit operators of structural learning, this means that neural creation is a much different problem from neural pruning. For the connection operators, because synaptogenesis occurs between existing structures, it is of the same search complexity as synaptic pruning, although metrics for where to add connections are not as straight-forward as where to remove them. Thus, synaptic pruning and synaptogenesis often look very similar, especially for ephemeral structural changes during training.

Nearly parallel to our pruning questions, our main questions for characterizing creation are:

- **Goals of Creation:** Why add connections or units to an ANN?
- **Architecture Specificity:** How has creation been adapted to different ANN architectures?
- **Measuring Entities:** How are the new entities to be added selected over other options?
- **Scheduling Creation:** When and how often does creation occur with respect to optimization of the existing parameters?
- **Operator Interactions:** What is the impact of using pruning along with creation, versus just one such modality?
- **Impact on Weight Optimization:** What is the impact of creation on optimization of parameter weights?

In discussing these questions, we focus on the growth aspects of the papers in our corpus: neural creation and synaptogenesis. We discuss the few topics that span both modalities of creation in Section 5.1, then cover each question specifically for synaptogenesis in Section 5.2 and for neural creation in Section 5.3 before concluding in Section 5.4.

5.1 Creation Generalities

Not as many shared characteristics exist between the modes of creation compared to that between the modes of pruning. We detail these few general characteristics below.

Goals of Creation: Growing in an ANN allows the architecture to be even more automatically customized, rather than having to hand-engineer the entire architecture or even the initial architecture before pruning. It may allow a much larger architectural search space to be explored over the course of structural learning, since this search space does not have to be explicitly predefined or instantiated. It is rather implicit from the algorithm's creation operations.

Impact on Weight Optimization: Creating entities in an ANN expands the dimensionality of the parameter search space. Network morphisms include ways of adding entities to an ANN with a weight initialization that preserves the functionality of the network (Wei et al., 2016). After using a network morphism to grow units or connections, the expanded ANN may then begin gradient descent at an equivalent location of the previous objective function space but now with expanded dimensions. This guarantees that the new optimal point will have at most the same cost if not lower, since the previous optimum will necessarily be included in the new search space within the subspace where all new parameters are nullified. However, adding too many dimensions increases the time and space costs for the ANN, including in search time if the search space becomes unnecessarily complex. Similarly as for pruning as discussed in section 4.1, creation is generally composed of greedy search decisions in a local search, which can only find local optima.

In the following subsection on synaptogenesis, we focus on algorithms that have a specific method for growing connections, rather than just reviving previously pruned connections, previously discussed as ephemeral pruning in Section 4.2, or creating units with their associated connections, to be discussed in Section 5.3.

5.2 Growing Connections

Goals of Creation: Synaptogenesis can be used in an ANN to create new connections between existing units. It has been used with pruning for discovering more effective flexible wirings between channels or neurons within ANNs, particularly without an initial overparameterized dense architecture (Puma-Villanueva et al., 2012; Bellec et al., 2018; Wortsman et al., 2019; Dai et al., 2019b;a). In other applications, Kim et al. (2021) augments an ANN's performance after training by adding neuron-level connections directly from hidden neurons to the output, while Elsken et al. (2017) searches for layer-level skip connections, among other network morphisms, concurrent with continual training during evolutionary architecture search.

Architecture Specificity: Neuron-level synaptogenesis algorithms can generally be applied in both convolutional and standard layers. For convolutional layers, synaptogenesis is generally done on the kernel level (Dai et al., 2019b;a; Wortsman et al., 2019). Only Bellec et al. (2018) performs synaptogenesis on individual parameters within each kernel, but does so randomly on masked connections to balance informed synaptic pruning. Wortsman et al. (2019) introduces a rewiring algorithm that is agnostic to whether connections are convolutional or dense and removes the constraint of layer organization that engineered deep architectures follow.

Measuring Entities: Metrics for synaptogenesis are less straightforward than for synaptic pruning, since the potential connections do not have active information flow. Thus, a common approach is to use a mask that inactivates connections during the forward pass but allows propagation of the gradient to them during the backward pass, providing many of the same metrics as are normally available (Dai et al., 2019b;a; Wortsman et al., 2019). To shrink the quadratic search space of all possible connections, Dai et al. (2019b;a) only allow connections between neurons in neighboring layers, while Wortsman et al. (2019) allows connections across multiple layers but still limited within modules of similar depth. Bellec et al. (2018) also uses a mask, but rather performs random synaptogenesis of a currently dormant connection each time another connection is pruned in order to maintain the sparsity level of the network. While masks bound the search space, that is not necessarily limiting to synaptogenesis since it already can only occur between existing units, whether or not a mask is used.

Beyond mask-based approaches, algorithms often use improved performance as the final metric for adding connections. Puma-Villanueva et al. (2012) ranks all possible connections with a mutual information heuristic amenable to synaptic creation and adds those that improve the performance. Kim et al. (2021) evaluates the effects on performance exhaustively for a smaller search space, only allowing hidden neurons to be connected directly to output neurons. Using improved performance as the selection criteria is intuitive, but is a greedy selection towards the goal of optimizing performance and is expensive to complete exhaustively without heuristics, limited search spaces, or randomized evaluations.

Scheduling Creation: Synaptogenesis is usually performed iteratively and greedily. New connections need to be optimized after a random initialization for optimal performance, so synaptogenesis phases are generally followed by weight optimization phases of either only the new parameters or the entire network.

Operator Interactions: Most synaptogenesis algorithms also implement synaptic pruning (Puma-Villanueva et al., 2012; Bellec et al., 2018; Dai et al., 2019b;a; Wortsman et al., 2019). Implementing synaptogenesis with synaptic pruning can create dynamically balanced connectivity, which may counteract any negative effects of earlier greedy synaptic additions to escape local optima. The remaining synaptogenesis algorithms only use synaptogenesis out of our two connection operators (Elsken et al., 2017; Kim et al., 2021). This simplifies the search process, but may approach local optima due to greediness.

Impact on Weight Optimization: Synaptogenesis adds dimensions to the search space. Because any added synaptic parameters can be set to zero to nullify their function and thus preserve the objective function of the ANN, synaptogenesis can guarantee the new global optimum has at most the same cost. However, depending on the initialization and training algorithms used, the synaptogenesis may displace the current location in the objective function space far enough to descend towards a different, possibly worse or better local optimum. Additionally, adding too many dimensions increases the time and space costs for the ANN, including in search time if the search space becomes unnecessarily complex.

5.3 Growing Units

Goals of Creation: Neuron creation, or adding units to an ANN, is the most open-ended neural operator. The general aims of neuron creation are to improve performance and to create an architecture that is specifically effective for the desired task without manual design. The resulting networks should have either reduced space and time complexity for deployment at a desired performance level or a better performance potential. Beyond algorithms with these general aims, some NAS approaches incorporate neural creation in the form of changing the architectural search space over the course of training (Laube & Zell, 2019; Ci et al., 2021; Roberts et al., 2021), which provides a larger search space than traditional pruning-only approaches. Both the general and NAS-specific applications allow for a more customized architecture with less hand-design.

Architecture Specificity: Neuron-level creation usually adds neurons of a predetermined type to existing or new layers. However, since selection techniques are often agnostic to layer type, many neural creation algorithms can be applied to architectures with layers of various types like convolutional or dense (Du et al., 2019; Liu et al., 2019a; Wu et al., 2020). Similarly to pruning, growing new filters in a convolutional layer also involves a mapping in the subsequent layer (Du et al., 2019), as shown in Figure 5. Layer-level creation often extends layers with the same type as previous layers (Dong et al., 2020; Wen et al., 2020). Roberts et al. (2021) avoids pre-engineering the layer type selection by parameterizing the layer function in a search space that includes most commonly used layer types in NAS among many others.

Measuring Entities: Selecting where and how to add new units to a network is not straight-forward. In order to keep the full search space unbounded, a mask cannot be used. Most approaches limit the instantaneous search space of possibilities, such as iteratively adding a single neuron to an existing layer (Ash, 1989; Fahlman & Lebiere, 1990; Frean, 1990; Lehtokangas, 1999; Ma & Khorasani, 2003; Islam et al., 2009; Dai et al., 2019b; Wu et al., 2020), adding a single new layer beyond the current architecture (Fahlman & Lebiere, 1990; Ma & Khorasani, 2003; Cortes et al., 2017; Wen et al., 2020; Wu et al., 2020), or splitting existing neurons (Du et al., 2019; Liu et al., 2019a; Wu et al., 2020). This results in a simpler problem at each step, but stills gives the algorithm the power of an unbounded search space over the course of training.

The simplest approaches, often combined with some form of pruning, use a generic creation scheme that adds some default number of new units (Narasimha et al., 2008; Gordon et al., 2018). Slightly more informed approaches use the network's objective function as a heuristical measure of success, adding the units that reduce the error the most out of generated options to the network (Cortes et al., 2017; Dai et al., 2019b).

Scheduling Creation: The objective function is used not only for creation selection but also for iterative scheduling and termination. In early construction algorithms, often a single neuron was added at a time upon convergence of the current network until the desired performance is reached (Ash, 1989; Fahlman & Lebiere, 1990; Frean, 1990; Lehtokangas, 1999; Ma & Khorasani, 2003; Islam et al., 2009; Puma-Villanueva et al., 2012). This approach of a single neuron added between training to convergence does not scale very well with the size of the network or complexity of the task. Many of the recent methods still use performance stagnation during training as a trigger for a neural creation phase, but for adding either many new neurons or an entire layer at a time Elsken et al. (2017); Cortes et al. (2017); Liu et al. (2019a). Otherwise, a manual schedule for iterative phases may be implemented (Narasimha et al., 2008; Du et al., 2019; Guo et al., 2021). Most of these schedules introduce hyperparameters, such as thresholds or durations per each phase of the schedule. These require hand-tuning or optimization for each specific dataset, task, and initial architecture. This is not desirable for generality and automation of algorithms across applications, but selecting hyperparameters is a much smaller search space for manual exploration than that of the architecture being automatically designed.

Operator Interactions: Most of the recent neural creation algorithms also use neural pruning, while the older ones did not. This dynamic approach not only can ameliorate effects of greedy structural changes, but also leaves the size of the final ANN as either an objective to optimize in tandem to performance or as an open-ended result. For example, Guo et al. (2021) can adapt initial seed networks to any computational budget, whether smaller, approximately the same, or larger than the original network. Connection operators may also be used in addition to unit operators to make finer adjustments to the network (Puma-Villanueva et al., 2012; Dai et al., 2019b; Du et al., 2019).

Impact on Weight Optimization: Neural creation expands the dimensionality of the search space of gradient descent. To determine the starting point in the newly expanded space, the weights of new units need to be initialized before they are trained. Unit-splitting algorithms (Lu et al., 2017; Du et al., 2019; Liu et al., 2019a; Dong et al., 2020; Wu et al., 2020) copy the weights and may add a small perturbation to ensure the units are not redundant if split in parallel. Network morphisms, used by (Islam et al., 2009; Elsken et al., 2017; Laube & Zell, 2019; Wen et al., 2020), initialize new structures to be null, which generally means initializing new weights to be zero. Most other neural creation algorithms use a similar initialization scheme to the original network, such as random weight initialization. Both unit-splitting without random noise and network morphisms perturb the location of the network in the new space less than methods that use random noise in their initialization. Both cases can be beneficial: staying in a similar location with newly added dimensions allows the path of gradient descent to continue with less disturbance or risk of regressing from the current level of performance, while a small perturbation may help escape local optima or saddle points (Liu et al., 2019a; Wu et al., 2020), as demonstrated in Figure 4.

5.4 Creation Conclusions

Creation has been less well-studied than pruning in ANNs, due to the additional complexity imposed by the open-ended search space of creation, which is less cooperative with back-propagation than static architectures and even pruning. Hand-designing neural architectures has also made it less practical and imperative, but the desire for it is continuing to increase alongside computational capabilities.

The brain has rather distinct phases of early development and adult life. These correspond roughly to neural architecture search space, or hand-design of an ANN architecture for the case of conventional non-structural learning, and the tuning of this architecture over the process of learning. Most structural learning research focuses on automating and improving the latter process, leaving further optimization of search space design as an open problem. However, the search space design is also synonymously influenced by the genetic priors used in the brain that have been optimized over the course of human evolution, making search space deign a much broader problem with a larger scope than one-shot learning.

The brain uses Hebbian learning, or correlations in activity between interacting neurons, to change effective connection strengths (Hebb, 1949). The explicit use of Hebbian learning in ANNs for weight training is, so far, not as popular or effective as back-propagation. Standard ANNs are not time-dependent, thus changing the nature of correlations in activity compared to that of the brain, and ANNs have global information access, permitting back-propagation of errors. However, Dai et al. (2019b) shows that growing connections via large values of the gradient of the network loss with respect to dormant masked weights is a hebbian-like rule. This suggests further exploration into appropriate application of Hebbian theory into artificial media.

As previously discussed in Section 4.4, the physical brain excels at local communication without indexing cost while ANNs can operate globally but with a cost per index. This has implications for effective creation in ANNs: non-random creation in ANNs, where each potential connection must be indexed and evaluated, should be instantaneously bounded to maintain search efficiency. Creation is not explicitly bounded in brains:

unit creation, including migration, is guided by genetic and developmental factors while synaptogenesis is guided locally by chemical signals. For synapse measurement, the brain does not use simulated heuristics like ANNs often do, but instead selects connections by dynamic processes of repeated synaptogenesis and synaptic removal in tandem with Hebbian learning. This suggests an approach for ANNs to make sub-optimal but less expensive additions balanced with more careful pruning to dynamically and efficiently construct an informed architecture. This method is already generally demonstrated in the brain, with higher neural creation rates with less synaptic guidance during early development then activity-driven dynamic synaptic connectivity and occurrence of dendritic synaptogenesis before firing action potentials in newly created neurons later in life.

6 Perspectives

With the deepened understanding of how each of the four neural operatorwork in ANNs, we present our perspectives on the overall state of structural learning. We first note implementational trends that shape current works in Section 6.1. We then discuss current challenges with future directions in Section 6.2.

6.1 Implementations and Trends

In order to successfully implement structural learning on modern machines rather than biological media, we note several approaches used commonly across algorithms. The ongoing development of computer hardware and software, which provide the media on which ANNs are implemented, has potentiated each of the innovations in ANNs throughout history. Present computational technology is particularly adept at handling array-based calculations. For example, GPUs drastically speed up ANN training wallclock time particularly with stochastic gradient descent (SGD) and tensor-based architecture parameterizations and data structures. However, they also impose a limit in random-access memory (RAM): ANN training is much more efficient and easier to implement if the entire memory-intensive backward pass of SGD can fit on the GPU's RAM at once. The inference computational cost of an ANN is often measured in the number of Multiply-Adds, or equivalently floating point operations (FLOPs), required per inference of a single input. This value can be used as part of a multi-objective approach, maximising performance while minimizing the inference cost. The training cost is usually measured in GPU-days, the number of days to train on a single GPU using current hardware and software. Low training costs are a key benefit of one-shot structural learning over techniques that instantiate and train many individual architectures. The following implementation trends are noted across our one-shot structural learning corpus, specified in Table 2.

Modularity: Dividing a design problem into multiple hierarchical levels makes it more tractable. For ANN architectures, this entails searching for smaller modular structures that can be used as building blocks to build larger architectures. This technique has already been used in the hand-design of architectures, like ResNet (He et al., 2016). It is also used in structural learning, notably unit creation and pruning where groups of parameters are added or removed as a unit. Modularity has especially been used in NAS in order to reduce the complexity of the search space. Most NAS algorithms that allow flexible interconnectivity use repeated cells of the same architecture during the search process. Some recent NAS approaches incorporate structural learning on multiple levels, such as searching for both layer-level and neuron-level structures (Yan et al., 2019). NAS algorithms incorporating creation take this even further by also optimizing the layer options used (Laube & Zell, 2019; Ci et al., 2021; Roberts et al., 2021), thus expanding the search space. Modularity is particularly capitalized in algorithms that either duplicate or split units (Lu et al., 2017; Du et al., 2019; Liu et al., 2019a; Dong et al., 2020; Wu et al., 2020). These algorithms reuse neurons, layers, or branches of the network that have already been trained on the present dataset and task as initialization points.

Masking and super-networks: A common approach to allow structural learning is to use masking within the ANN implementation. This mask is an encoding of the structure, enumerating all possible entities and controlling which ones are actively used in a given structure. By relaxing this mask to be continuous, the mask can permit continuous learning where the mask itself is treated like parameters of the network that can be optimized and discretized, such as in pruning or continuous neural architecture search.

When implemented on the layer level for NAS, masking also naturally leads to weight-sharing, where many different architectures are represented with shared parameter data structures but the mask imposes different functionality by specifying the architectural path and thus the parameters used within the super-network. The architectures parameters used for this specification in continuous NAS are usually activated by softmax, gumbel-softmax, or another balancing function in order to enforce scaling across parallel options. Gumbel-softmax and annealing are more recent innovations in activation of NAS parameters to smooth the transition to the final discretization (Guo et al., 2020; Kang & Han, 2020; Noy et al., 2020; Wan et al., 2020). Other NAS papers have introduced non-competitive activation functions (Chu et al., 2020). Once activated, the architecture parameters may then be used for weighted sums of the activations from parallel layer options in continuous architecture search. For path-sampling architecture search, the activated parameters are often used as a sampling probability.

When implemented on the neuron level, masking for synaptic pruning, such as in unstructured pruning algorithms, allows for more native linear algebra, but does not lead to any computational speed-up without specialized software. Because masks for neural pruning control multiple synaptic parameters per unit with a single masking parameter, they may require regularization to enforce group-wise operations on the parameters during the algorithm, or else the mask may be implemented across dimensions of the parameter matrices if that is compatible with the parameter data structure representation, such as batchnorm scaling factors (Gordon et al., 2018). Like NAS, some neuron-level masking implementations also use temperature in their implementation to smooth the transition from the continuous relaxation to discrete masking (Bellec et al., 2018).

Masking naturally turns architectural selection problems into a pruning problem with a bounded search space. This often creates a limitation in neural architecture search and pruning methods where these methods are used: the maximum size network over the course of search should fit its forward and backward passes on a GPU at once, even if the desired final network will be much smaller. For example, super-networks in NAS are often limited by the GPU size: for continuous NAS methods, all options are at least partially activated, resulting in a very large network to evaluate and optimize. In order for the final network discretized from the super-network to also approach the GPU size, many continuous NAS works search in a shallower architecture with fewer channels and then implement the discretized structure in a deeper architecture by repeating cells, which is another benefit of using modular cells. However, this restrains the architecture search to repeatable cells with channels, which is not always applicable for tasks outside of image classification. It also further deepens the discretization gap by contributing to the structural differences between the super-network and the discretized network, as discussed in Section 4.3.

Dynamically sparse training methods avoid the large initial model by sparsely initializing the network and dynamically adjusting which entities are active, allowing the network to maintain a maximum density level (Bellec et al., 2018; Wortsman et al., 2019). This results in a predefined structural search space without limitations on the size of the full search space, but requires structural learning processes that are not memory-bound to fully reap the benefits over standard pruning techniques.

Interconnectivity: The brain shows connectivity patterns that are much more flexible and complex than modern standard feed-forward networks, where neurons are organized in layers and each layer is only connected to its immediate neighbors. Neural architectures tended towards this organized structure in the early decades of backpropagation, particularly for the multi-layer perceptron. Recently, skip-connections broke this simple patterning and improved the state of the art in computer vision and many other tasks (He et al., 2016). It allows for much deeper networks to be trained without vanishing gradients and for higher and lower-level features to be used together. Many structural learning algorithms like NAS allow for interconnectivity flexibility on the layer level, while Bellec et al. (2018); Wortsman et al. (2019) allow even finer interconnectivity outside of layer organization on the neuron level. Including the potential for such flexible interconnectivity in structural learning, rather than using a rigid connectivity backbone, greatly expands the search space and thus adds complexity to the search, but also yields the potential for more powerful networks.

Weight Reinitialization: Most neural-level structural learning algorithms are end-to-end, where weights learned in tandem to the architecture are used in the final ANN either without any further training or with only finetuning. In contrast, most layer-level continuous NAS approaches require full weight reinitialization and training after architecture discretization. While this does allow the super-network and discretized architecture to have different shapes and training hyperparameters, it is generally more expensive and has a risk of uncorrelated performance between the two stages. This trend is also seen in Frankle & Carbin (2018), although here the pruned network returns to the initial initialization for the remaining weights.

6.2 Current Challenges and Future Directions for Development

As computational speed and power continues to increase, the desire to automate the engineering of ANN architectures is increasing as well. It will be a crucial step to continue in the direction towards general intelligence. Human engineering is currently a rate-limiting step of ANN architecture innovation. Among discussions already presented throughout this paper, we will end our discussion with present challenges and future directions for development.

Many of the benchmark applications that structural learning algorithms are implemented and demonstrated for are well-studied supervised tasks over datasets with static and representative distributions across the training and test partitions. While these provide a consistent benchmark against hand-designed networks and are useful for algorithm development, their utilization is almost paradoxical for one of the powerful potentials of structural learning: automating structural design even for tasks that are not well-studied. Structural learning may allow for more power in less studied and engineered applications, bypassing the need for specific architectural innovations to be designed by hand for every such task. This phenomenon is particularly noted in NAS; automating the engineering of architectures could potentiate the application of ANNs to less-studied tasks, but the majority of NAS papers only report results on the same few image classification datasets for which architectures have already been thoroughly researched, each trial trained independently from scratch, with image-specific search spaces like convolutional layers in repeatable cells. Learning environments that go beyond the basics include multi-task learning (Lu et al., 2017; Dai et al., 2019a; Guo et al., 2020; Peng et al., 2021), continual learning (Dai et al., 2019a; Li et al., 2019; Wu et al., 2020; Peng et al., 2021; Niu et al., 2021), and reinforcement learning (Hornby & Pollack, 2001; Fu et al., 2020; Miao et al., 2021). These may also approach the dynamic learning environment of the human brain more closely, where representations, labels, and rewards must be internally implemented (Botvinick et al., 2020; Banerjee et al., 2021). Few structural learning papers have implemented their algorithms in these environments thus far, but they represent a deeper level of artificial intelligence.

Our definition of structural learning implies dimensionality changes to the objective function space of ANNs. Structural optimization alongside weight optimization thus prompts a multi-level optimization, with the structure of the network being the outer-most level that defines the space of parameters being optimized on a lower level (Xie et al., 2021). This is explicitly considered in many continuous super-network NAS papers as a bilevel problem (Liu et al., 2018), but the optimization is often simplified to alternating gradient descent steps without theoretical guarantees but with reasonable performance relative to time and computation cost in practice. Multi-level optimization methods are continuously being developed (Huang & Huang, 2021) and may be applied to structural learning, but the trade-off of performance with complexity cost must be considered for effective use.

The level of abstraction, demonstrated in Figure 1, has resulting trade-offs for the structural learning algorithm employing it. Lower level abstractions, such as the direct paradigm of feed-forward neurons, allow for fine-grained tuning of the architecture, but may be too minute for architectures with multiple higher orders of structure and billions of parameters. Higher level paradigms, towards NAS super-networks, impose difficulty by requiring additional structural design in the search space, such as selecting or even structurally optimizing the layer options present in each connection. Combining multiple levels within a single algorithm has thus far been rare (Yan et al., 2019), but presents a powerful direction for structural learning, requiring coordination of each level that has only been studied in isolation thus far.

Structural creation is the most difficult form of structural learning because it is open-ended and slow with current methods. However, such a large search space gives it the potential to be the most fruitful in the pursuit of automating architectural design of ANNs. It is represented not only in selecting where and how to add new units, but also in widening the search spaces of potential layer types and connectivity patterns, which are currently rather limited to discrete, hand-engineered choices. In order to surpass this limited space, Bronstein et al. (2021) presents a geometric framework to unite diverse ANN architectures, layer types, and data structure types. Applying these ideas to ANN structural learning could expand the search space by utilizing symmetry-inducing invariances with less restrictions on the input data structure, thus supporting applications even to less naturally tensor-like data structures.

In biology, the genetic code potentiates each neural operatorwithin biological media. In ANNs, the algorithm can be considered like the genetic sequence which determine exactly how ANN learning, both basic weight optimization and the more difficult structural learning, occurs within the computational media of hardware and software given an individual in a learning scenario with training data. The developmental search space is encoded in the algorithm. Most such genetic sequences are being hand-designed, but some efforts have been made towards recipe search, where the algorithm itself is optimized. Evolution was the crucial process for discovering the current foundation for biological structural learning, such as defining the phases of high neurogenesis in early development to dynamic connectivity later in life. Some efforts towards replicating this process are in evolutionary recipe search, where an evolutionary algorithm tries to optimize the entire procedure of structuring and training a neural network. This represents learning on multiple timescales, beyond just the lifetime of a single ANN.

Beyond providing a biologically-inspired framework for structural learning in ANNs, we also intend for this work to strengthen connections between the various sub-communities performing structural learning. We noted a lack of citations both across time and across sub-communities. Many pruning algorithms in the last five years implement the same main abstract methodology as those from three decades ago (Blalock et al., 2020), but now show drastic improvements in performance. While pruning and NAS works are somewhat well-connected within their respective corpora, developmental works using the creation operations are not, due to both a lack of common vocabulary and a relatively low frequency of new methods. Our provided framework can help bridge the abstract similarities between algorithms to promote collaboration, not only between the neuroscience and machine learning communities, but also between the sub-communities of the latter.

6.3 Conclusion

Structural learning in ANNs is a dynamic and diverse field with a vast potential. Our biologically-inspired framework of neural operators, consisting of neural creation, synaptogenesis, neural pruning, and synaptic pruning, provides a means for synergy not only between neuroscience and machine learning, but also between subcommunities within structural machine learning such as pruning, AutoML, neural architecture search, and developmental neural networks. Furthering structural learning across these approaches could lead to significant breakthroughs beyond the current state of the art in artificial intelligence and machine learning.

References

Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas Donald Lane. Zero-cost proxies for lightweight nas. In *International Conference on Learning Representations*, 2020.

Sandra Ackerman. The development and shaping of the brain. *Discovering the Brain*, pp. 86–103, 1992.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 139–154, 2018.

Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(3):32, 2017.

Timur Ash. Dynamic Node Creation in Backpropagation Networks. *Connection Science*, 1(4):365–375, jan 1989. ISSN 13600494. doi: 10.1080/09540098908915647.

Frederico AC Azevedo, Ludmila RB Carvalho, Lea T Grinberg, José Marcelo Farfel, Renata EL Ferretti, Renata EP Leite, Wilson Jacob Filho, Roberto Lent, and Suzana Herculano-Houzel. Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, 513(5):532–541, 2009.

Lei Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pp. 2654–2662, 2014.

Abhishek Banerjee, Rajeev V Rikhye, and Adam Marblestone. Reinforcement-guided learning in frontal neocortex: emerging computational concepts. *Current Opinion in Behavioral Sciences*, 38:133–140, 2021.

Brian Bartoldson, Ari Morcos, Adrian Barbu, and Gordon Erlebacher. The generalization-stability tradeoff in neural network pruning. *Advances in Neural Information Processing Systems*, 33:20852–20864, 2020.

Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. *International Conference on Learning Representations*, 2018.

Kaifeng Bi, Lingxi Xie, Xin Chen, Longhui Wei, and Qi Tian. GOLD-NAS: Gradual, One-Level, Differentiable. *arXiv preprint arXiv:2007.03331*, 2020.

Yijun Bian, Qingquan Song, Mengnan Du, Jun Yao, Huanhuan Chen, and Xia Hu. Subarchitecture ensemble pruning in neural architecture search. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

Derron L Bishop, Thomas Misgeld, Mark K Walsh, Wen-Biao Gan, and Jeff W Lichtman. Axon branch removal at developing synapses by axosome shedding. *Neuron*, 44(4):651–661, 2004.

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020.

Matthew Botvinick, Jane X Wang, Will Dabney, Kevin J Miller, and Zeb Kurth-Nelson. Deep reinforcement learning and its neuroscientific implications. *Neuron*, 2020.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Robert R Buss, Woong Sun, and Ronald W Oppenheim. Adaptive roles of programmed cell death during nervous system development. *Annu. Rev. Neurosci.*, 29:1–35, 2006.

Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2019.

Shuxin Chen, Lin Lin, Zixun Zhang, and Mitsuo Gen. Evolutionary netarchitecture search for deep neural networks pruning. In *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, pp. 189–196, 2019.

Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair darts: Eliminating unfair advantages in differentiable architecture search. In *European Conference on Computer Vision*, pp. 465–480. Springer, 2020.

Yuanzheng Ci, Chen Lin, Ming Sun, Boyu Chen, Hongwen Zhang, and Wanli Ouyang. Evolving search space for neural architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6659–6669, 2021.

Corinna Cortes, Xavier Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. Adanet: Adaptive structural learning of artificial neural networks. In *International conference on machine learning*, pp. 874–883. PMLR, 2017.

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

Xiaoliang Dai, Hongxu Yin, and Niraj K. Jha. Incremental Learning Using a Grow-and-Prune Paradigm with Efficient Neural Networks. *IEEE Transactions on Emerging Topics in Computing*, may 2019a. ISSN 21686750. doi: 10.1109/TETC.2020.3037052.

Xiaoliang Dai, Hongxu Yin, and Niraj K Jha. NeST: A neural network synthesis tool based on a grow-and-prune paradigm. *IEEE Transactions on Computers*, 68(10):1487–1497, 2019b.

Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4):485–532, 2020.

Chengyu Dong, Liyuan Liu, Zichao Li, and Jingbo Shang. Towards adaptive residual network training: A neural-ode perspective. In *International Conference on Machine Learning*, pp. 2616–2626. PMLR, 2020.

Xiaocong Du, Zheng Li, Yufei Ma, and Yu Cao. Efficient network construction through structural plasticity. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(3):453–464, 2019.

Thomas Elsken, Jan-Hendrik Metzen, and Frank Hutter. Simple and efficient architecture search for convolutional neural networks. In *NIPS Workshop on Meta-Learning*, 2017.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20:1–21, 2019.

Aurélie Ernst, Kanar Alkass, Samuel Bernard, Mehran Salehpour, Shira Perl, John Tisdale, Göran Possnert, Henrik Druid, and Jonas Frisén. Neurogenesis in the striatum of the adult human brain. *Cell*, 156(5): 1072–1083, 2014.

Lynda Erskine and Eloisa Herrera. The retinal ganglion cell axon's journey: Insights into molecular mechanisms of axon guidance. *Developmental biology*, 308(1):1–14, 2007.

Scott E Fahlman and Christian Lebiere. The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems*, pp. 524–532, 1990.

Travis E Faust, Georgia Gunner, and Dorothy P Schafer. Mechanisms governing activity-dependent synaptic pruning in the developing mammalian cns. *Nature Reviews Neuroscience*, pp. 1–17, 2021.

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

Marcus Frean. The upstart algorithm: A method for constructing and training feedforward neural networks. *Neural computation*, 2(2):198–209, 1990.

Michael Fricker, Aviva M Tolkovsky, Vilmante Borutaite, Michael Coleman, and Guy C Brown. Neuronal cell death. *Physiological reviews*, 98(2):813–880, 2018.

Yonggan Fu, Zhongzhi Yu, Yongan Zhang, and Yingyan Lin. Auto-agent-distiller: Towards efficient deep reinforcement learning agents via neural architecture search. *arXiv preprint arXiv:2012.13091*, 2020.

H Troy Ghashghaei, Cary Lai, and ES Anton. Neuronal migration in the adult brain: are we there yet? *Nature Reviews Neuroscience*, 8(2):141–151, 2007.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Hao Wu, Tien-Ju Yang, and Edward Choi. Morphnet: Fast & simple resource-constrained structure learning of deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1586–1595, 2018.

Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. Learning to branch for multi-task learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 3854–3863. PMLR, 13–18 Jul 2020.

Shaopeng Guo, Yujie Wang, Kun Yuan, and Quanquan Li. Differentiable network adaption with elastic search space. *arXiv preprint arXiv:2103.16350*, 2021.

Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. *Advances in neural information processing systems*, 29, 2016.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

Anthony W Harrington and David D Ginty. Long-distance retrograde neurotrophic factor signalling in neurons. *Nature Reviews Neuroscience*, 14(3):177–187, 2013.

Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pp. 293–299. IEEE, 1993.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

Donald Olding Hebb. *The organisation of behaviour: a neuropsychological theory.* Science Editions New York, 1949.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

Gregory S Hornby and Jordan B Pollack. Body-brain co-evolution using L-systems as a generative encoding. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, pp. 868–875, 2001.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Feihu Huang and Heng Huang. Biadam: Fast adaptive bilevel optimization methods. *arXiv preprint arXiv:2106.11396*, 2021.

David Hunter Hubel, Torsten Nils Wiesel, Simon LeVay, Horace Basil Barlow, and Raymond Michael Gaze. Plasticity of ocular dominance columns in monkey striate cortex. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 278(961):377–409, 1977.

Md Monirul Islam, Md Abdus Sattar, Md Faijul Amin, Xin Yao, and Kazuyuki Murase. A new constructive algorithm for architectural and functional adaptation of artificial neural networks. *IEEE transactions on systems, man, and cybernetics, part B (cybernetics)*, 39(6):1590–1605, 2009.

Minsoo Kang and Bohyung Han. Operation-aware soft channel pruning using differentiable masks. In *International Conference on Machine Learning*, pp. 5122–5131. PMLR, 2020.

Larry C Katz and Carla J Shatz. Synaptic activity and the construction of cortical circuits. *Science*, 274 (5290):1133–1138, 1996.

Wolfgang Kelsch, Shuyin Sim, and Carlos Lois. Watching synaptogenesis in the adult brain. *Annual review of neuroscience*, 33:131–149, 2010.

Jinwook Kim, Min-Soo Kim, and Heeyong Yoon. Tweaking deep neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Kevin Alexander Laube and Andreas Zell. Prune and replace nas. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 915–921. IEEE, 2019.

César Laurent, Camille Ballas, Thomas George, Nicolas Ballas, and Pascal Vincent. Revisiting loss modelling for unstructured pruning. *arXiv preprint arXiv:2006.12279*, 2020.

Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, pp. 598–605, 1990.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2019.

Mikko Lehtokangas. Modelling with constructive backpropagation. *Neural Networks*, 12(4-5):707–716, 1999.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *International Conference on Learning Representations*, 2017.

Ping Li, Igor Farkas, and Brian MacWhinney. Early lexical development in a self-organizing neural network. *Neural Networks*, 17(8):1345–1362, 2004. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2004.07.004. New Developments in Self-Organizing Systems.

Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning*, pp. 3925–3934. PMLR, 2019.

Grace W Lindsay. Attention in psychology, neuroscience, and machine learning. *Frontiers in computational neuroscience*, pp. 29, 2020.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

Qiang Liu, Lemeng Wu, and Dilin Wang. Splitting steepest descent for growing neural architectures. *Advances in neural information processing systems*, 2019a.

Risheng Liu, Jiaxin Gao, Jin Zhang, Deyu Meng, and Zhouchen Lin. Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021a.

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Zahra Atashgahi, Lu Yin, Huanyu Kou, Li Shen, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. Sparse training via boosting pruning plasticity with neuroregeneration. *Advances in Neural Information Processing Systems*, 34, 2021b.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019b.

Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5334–5343, 2017.

Zhichao Lu, Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. Nsganetv2: Evolutionary multi-objective surrogate-assisted neural architecture search. In *European Conference on Computer Vision*, pp. 35–51. Springer, 2020.

Liying Ma and Khashayar Khorasani. A new strategy for adaptively constructing multilayer feedforward neural networks. *Neurocomputing*, 51:361–385, 2003.

Kaitlin Maile, Erwan Lecarpentier, Hervé Luga, and Dennis G Wilson. Darts-prime: Regularization and scheduling improve constrained optimization in differentiable nas. *arXiv preprint arXiv:2106.11655*, 2021.

Sabrina Malik, Govindaiah Vinukonda, Linnea R Vose, Daniel Diamond, Bala BR Bhimavarapu, Furong Hu, Muhammad T Zia, Robert Hevner, Nada Zecevic, and Praveen Ballabh. Neurogenesis continues in the third trimester of pregnancy and is suppressed by premature birth. *Journal of neuroscience*, 33(2): 411–423, 2013.

Charles E Martin and Praveen K Pilly. Probabilistic Program Neurogenesis. In *The 2018 Conference on Artificial Life: A Hybrid of the European Conference on Artificial Life (ECAL) and the International Conference on the Synthesis and Simulation of Living Systems (ALIFE)*, pp. 440–447. MIT Press, 2019.

Jieru Mei, Yingwei Li, Xiaochen Lian, Xiaojie Jin, Linjie Yang, Alan Yuille, and Jianchao Yang. Atomnas: Fine-grained end-to-end neural architecture search. In *International Conference on Learning Representations*, 2019.

Hrushikesh N Mhaskar and Tomaso Poggio. Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*, 14(06):829–848, 2016.

Yingjie Miao, Xingyou Song, Daiyi Peng, Summer Yue, Eugene Brevdo, and Aleksandra Faust. Rl-darts: Differentiable architecture search for reinforcement learning. *arXiv preprint arXiv:2106.02229*, 2021.

Julian Francis Miller. Improbed: Multiple problem-solving brain via evolved developmental programs. *Artificial Life*, 27(3–4):300–335, 2022.

Guo-li Ming and Hongjun Song. Adult neurogenesis in the mammalian brain: Significant answers and significant questions. *Neuron*, 70(4):687–702, 2011.

Riccardo Mozzachiodi and John H Byrne. More than synaptic plasticity: role of nonsynaptic plasticity in learning and memory. *Trends in neurosciences*, 33(1):17–26, 2010.

Pramod L Narasimha, Walter H Delashmit, Michael T Manry, Jiang Li, and Francisco Maldonado. An integrated growing-pruning method for feedforward network training. *Neurocomputing*, 13(71):2831–2847, 2008.

Thomas J Nelson and Daniel L Alkon. Molecular regulation of synaptogenesis during associative learning and memory. *Brain research*, 1621:239–251, 2015.

Shuaicheng Niu, Jiaxiang Wu, Guanghui Xu, Yifan Zhang, Yong Guo, Peilin Zhao, Peng Wang, and Mingkui Tan. Adaxpert: Adapting neural architecture for growing data. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8184–8194. PMLR, 18–24 Jul 2021.

Asaf Noy, Niv Nayman, Tal Ridnik, Nadav Zamir, Sivan Doveh, Itamar Friedman, Raja Giryes, and Lihi Zelnik. Asap: Architecture search, anneal and prune. In *International Conference on Artificial Intelligence and Statistics*, pp. 493–503. PMLR, 2020.

Wei Pan, Hao Dong, and Yike Guo. DropNeuron: Simplifying the Structure of Deep Neural Networks. *arXiv:1606.07326 [cs, stat]*, jul 2016.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

Jian Peng, Bo Tang, Hao Jiang, Zhuo Li, Yinjie Lei, Tao Lin, and Haifeng Li. Overcoming long-term catastrophic forgetting through adversarial neural pruning and synaptic consolidation. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

Cory Pfeiffenberger, Tyler Cutforth, Georgia Woods, Jena Yamada, René C Rentería, David R Copenhagen, John G Flanagan, and David A Feldheim. Ephrin-As and neural activity are required for eye-specific patterning during retinogeniculate mapping. *Nature neuroscience*, 8(8):1022, 2005.

Tomaso Poggio, Hrushikesh Mhaskar, Lorenzo Rosasco, Brando Miranda, and Qianli Liao. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*, 14(5):503–519, 2017.

Wilfredo J. Puma-Villanueva, Eurípedes P. dos Santos, and Fernando J. Von Zuben. A constructive algorithm to synthesize arbitrarily connected feedforward neural networks. *Neurocomputing*, 75(1):14–32, jan 2012. ISSN 18728286. doi: 10.1016/j.neucom.2011.05.025.

Philip T Quinlan. Structural change and development in real and artificial neural networks. *Neural Networks*, 11(4):577–599, 1998.

Brian G Rash, James B Ackman, and Pasko Rakic. Bidirectional radial ca2+ activity regulates neurogenesis and migration during early cortical column formation. *Science advances*, 2(2):e1501733, 2016.

Martin M Riccomagno and Alex L Kolodkin. Sculpting neural circuits by axon and dendrite pruning. *Annual review of cell and developmental biology*, 31:779–805, 2015.

Nicholas Roberts, Mikhail Khodak, Tri Dao, Liam Li, Christopher Ré, and Ameet Talwalkar. Rethinking neural operations for diverse tasks. *Advances in Neural Information Processing Systems*, 34, 2021.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks. *arXiv:1606.04671 [cs]*, September 2016.

Charles Siegel, Jeff Daily, and Abhinav Vishnu. Adaptive neuron apoptosis for accelerating deep learning on large scale systems. In *2016 IEEE International Conference on Big Data (Big Data)*, pp. 753–762. IEEE, 2016.

Nilotpal Sinha and Kuan-Wen Chen. Evolving neural architecture using one shot model. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 910–918, 2021.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15 (1):1929–1958, 2014.

Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

Kenneth O Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1(1):24–35, 2019.

Beth Stevens, Nicola J Allen, Luis E Vazquez, Gareth R Howell, Karen S Christopherson, Navid Nouri, Kristina D Micheva, Adrienne K Mehalow, Andrew D Huberman, Benjamin Stafford, et al. The classical complement cascade mediates cns synapse elimination. *Cell*, 131(6):1164–1178, 2007.

Noelia Urbán and François Guillemot. Neurogenesis in the embryonic and adult brain: same regulators, different roles. *Frontiers in cellular neuroscience*, 8:396, 2014.

Pierre Vanderhaeghen and Hwai-Jong Cheng. Guidance molecules in axon pruning and cell death. *Cold Spring Harbor perspectives in biology*, 2(6):a001859, 2010.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Tom Veniat and Ludovic Denoyer. Learning time/memory-efficient deep architectures with budgeted super networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3492–3500, 2018.

Clarissa L Waites, Ann Marie Craig, and Craig C Garner. Mechanisms of vertebrate synaptogenesis. *Annu. Rev. Neurosci.*, 28:251–274, 2005.

Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12965–12974, 2020.

Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable nas. In *International Conference on Learning Representations*, 2021.

Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. Network morphism. In *International Conference on Machine Learning*, pp. 564–572. PMLR, 2016.

Wei Wen, Feng Yan, Yiran Chen, and Hai Li. AutoGrow: Automatic Layer Growing in Deep Convolutional Networks. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 20:833–841, 2020. doi: 10.1145/3394486.3403126.

Daniel K Wilton, Lasse Dissing-Olesen, and Beth Stevens. Neuron-glia signaling in synapse elimination. *Annual review of neuroscience*, 42:107–127, 2019.

Mitchell Wortsman, Ali Farhadi, and Mohammad Rastegari. Discovering neural wirings. *Advances in Neural Information Processing Systems*, 32:2684–2694, 2019.

Lemeng Wu, Bo Liu, Peter Stone, and Qiang Liu. Firefly Neural Architecture Descent: a General Approach for Growing Neural Networks. In *Advances in Neural Information Processing Systems*, 2020.

Wei Wu, Kit Wong, Jin-hui Chen, Zhi-hong Jiang, Sophie Dupuis, Jane Y Wu, and Yi Rao. Directional guidance of neuronal migration in the olfactory system by the protein slit. *Nature*, 400(6742):331–336, 1999.

Lingxi Xie, Xin Chen, Kaifeng Bi, Longhui Wei, Yuhui Xu, Lanfei Wang, Zhengsu Chen, An Xiao, Jianlong Chang, Xiaopeng Zhang, et al. Weight-sharing neural architecture search: A battle to shrink the optimization gap. *ACM Computing Surveys (CSUR)*, 54(9):1–37, 2021.

Shen Yan, Biyi Fang, Faen Zhang, Yu Zheng, Xiao Zeng, Mi Zhang, and Hui Xu. Hm-nas: Efficient neural architecture search via hierarchical masking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019.

Quanming Yao, Ju Xu, Wei-Wei Tu, and Zhanxing Zhu. Efficient neural architecture search via proximal iterations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Mao Ye, Chengyue Gong, Lizhen Nie, Denny Zhou, Adam Klivans, and Qiang Liu. Good subnetworks provably exist: Pruning via greedy forward selection. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10820–10830. PMLR, 13–18 Jul 2020.

Weeteck Yeo and Jean Gautier. Early neural cell death: dying to become neurons. *Developmental biology*, 274(2):233–244, 2004.

Yoshihiro Yoshihara, Mathias De Roo, and Dominique Muller. Dendritic spine formation and stabilization. *Current opinion in neurobiology*, 19(2):146–153, 2009.

Xin Yuan, Pedro Henrique Pamplona Savarese, and Michael Maire. Growing efficient deep networks by structured continuous sparsification. In *International Conference on Learning Representations*, 2021.

Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Gradient descent optimizes over-parameterized deep relu networks. *Machine Learning*, 109(3):467–492, 2020.

	Operator type				Unit type				Layer type			Task					
Reference	Neural creation	Synaptogenesis	Neural pruning	Synaptic pruning	Neuron	Kernel shape	Filter/channel	Layer(s)	Dense	Conv/pooling	Recurrent	Multi-task	Continual	Image classification	Other computer vision	NLP/sequence	Supervised learning
Ash (1989)	+				+				+								+
Fahlman & Lebiere (1990)	+				+			+	+								+
Frean (1990)	+				+				+								+
LeCun et al. (1990)				+	+				+					+			+
Hassibi et al. (1993)				+	+				+								+
Lehtokangas (1999)	+				+				+								+
Ma & Khorasani (2003)	+				+			+	+								+
Narasimha et al. (2008)	+		+		+				+								+
Islam et al. (2009)	+				+			+	+								+
Puma-Villanueva et al. (2012)	+	+	+	+	+				+								+
Han et al. (2015)				+	+				+	+				+			+
Guo et al. (2016)				+	+		+		+	+				+			+
Pan et al. (2016)			+	+	+				+								+
Siegel et al. (2016)			+		+				+	+				+			+
Anwar et al. (2017)			+	+		+	+			+				+			+
Cortes et al. (2017)	+		+					+	+								+
Elsken et al. (2017)	+	+				+	+	+		+				+			+
Li et al. (2017)			+				+			+				+			+
Lu et al. (2017)	+							+	+	+		+		+			+
Bellec et al. (2018)		+		+	+					+	+			+		+	+
Gordon et al. (2018)	+		+		+		+		+	+				+			+
Liu et al. (2018)			+					+		+	+			+		+	+
Veniat & Denoyer (2018)			+				+	+		+				+	+		+
Cai et al. (2019)			+					+		+				+			+
Chen et al. (2019)			+				+	+		+				+			+
Dai et al. (2019a)		+		+	+		+		+	+	+	+	+	+		+	+
Dai et al. (2019b)	+	+	+	+	+		+			+				+			+
Du et al. (2019)	+		+	+	+		+		+	+				+			+
Laube & Zell (2019)	+		+			+	+	+		+				+			+
Lee et al. (2019)				+	+		+		+	+	+			+		+	+
Li et al. (2019)			+					+	+	+			+	+			+
Liu et al. (2019a)	+				+		+		+	+				+		+	+
Mei et al. (2019)			+			+	+	+		+				+			+
Wortsman et al. (2019)		+		+			+			+				+			+
Yan et al. (2019)			+	+	+			+		+				+			+
Bi et al. (2020)			+					+		+				+			+
Dong et al. (2020)	+							+		+				+			+
Guo et al. (2020)			+	+				+	+	+		+		+	+		+
Kang & Han (2020)			+				+			+							+
Noy et al. (2020)			+					+		+				+			+
Wan et al. (2020)			+			+	+			+				+			+
Wen et al. (2020)	+							+		+				+			+
Wu et al. (2020)	+		+		+		+	+	+	+			+	+			+
Yao et al. (2020)			+					+		+	+			+		+	+
Ye et al. (2020)			+		+		+		+	+				+			+
Bian et al. (2021)			+					+	+	+				+			+
Ci et al. (2021)			+			+		+		+				+			+
Guo et al. (2021)	+		+				+	+		+				+			+
Kim et al. (2021)		+			+				+	+				+			+
Peng et al. (2021)				+	+				+	+		+	+	+	+		+
Roberts et al. (2021)	+		+			+	+	+		+	+			+		+	+
Sinha & Chen (2021)			+					+		+				+			+
Wang et al. (2021)				+				+		+				+			+
Yuan et al. (2021)			+				+			+	+			+	+	+	+

Table 1: Operator type, unit type, layer type, and task qualities across the collected corpus.

	Method										Implementation			
Reference	Mask descent	RL generator	Evolutionary	Magnitude-based	1st order metric	2nd order metric	Other metric-based	Improvement-based	Regularization	Splitting/joining	Modularity	Weight-share/Mask	Interconnectivity	Final reinitialization
Ash (1989)								+						
Fahlman & Lebiere (1990)								+						
Frean (1990)								+						
LeCun et al. (1990)						+								
Hassibi et al. (1993)						+								
Lehtokangas (1999)								+						
Ma & Khorasani (2003)								+						
Narasimha et al. (2008)							+	+						
Islam et al. (2009)								+						
Puma-Villanueva et al. (2012)								+					+	
Han et al. (2015)				+					+			+		
Guo et al. (2016)				+								+		
Pan et al. (2016)				+					+					
Siegel et al. (2016)										+				
Anwar et al. (2017)			+									+		
Cortes et al. (2017)			+										+	
Elsken et al. (2017)			+									+	+	
Li et al. (2017)				+										
Lu et al. (2017)									+	+		+		
Bellec et al. (2018)	+			+					+					
Gordon et al. (2018)									+					
Liu et al. (2018)	+			+							+	+	+	+
Veniat & Denoyer (2018)	+				+				+			+	+	+
Cai et al. (2019)	+	+		+							+	+		+
Chen et al. (2019)			+									+		
Dai et al. (2019a)				+	+							+		
Dai et al. (2019b)				+	+									
Du et al. (2019)					+					+				
Laube & Zell (2019)	+				+					+	+	+	+	+
Lee et al. (2019)					+									
Li et al. (2019)	+			+								+		
Liu et al. (2019a)							+			+				
Mei et al. (2019)	+				+						+	+		
Wortsman et al. (2019)	+			+								+	+	
Yan et al. (2019)	+			+							+	+	+	
Bi et al. (2020)	+			+							+	+	+	+
Dong et al. (2020)							+							
Guo et al. (2020)					+							+		+
Kang & Han (2020)	+				+				+			+		
Noy et al. (2020)	+			+							+	+	+	+
Wan et al. (2020)	+			+								+		+
Wen et al. (2020)								+						
Wu et al. (2020)					+					+				
Yao et al. (2020)	+			+							+	+	+	+
Ye et al. (2020)								+				+		
Bian et al. (2021)	+						+	+					+	
Ci et al. (2021)	+		+	+								+	+	+
Guo et al. (2021)	+			+								+		+
Kim et al. (2021)								+					+	
Peng et al. (2021)					+							+		
Roberts et al. (2021)	+			+							+	+		
Sinha & Chen (2021)			+									+	+	+
Wang et al. (2021)	+							+				+		+
Yuan et al. (2021)	+			+								+		

Table 2: Method and implementation qualities across the collected corpus.