Creating, Managing, and Understanding Large, Sparse, Multitask Neural Networks

Harshvardhan Sikka
has727@g.harvard.edu

1 Introduction

One of the popular directions in Deep Learning (DL) research has been to build larger and more complex deep networks that can perform well on several different learning tasks, commonly known as multitask learning. This work is usually done within specific domains, e.g. multitask models that perform captioning, translation, and text classification tasks. Some work has been done in building multimodal/crossmodal networks that use deep networks with a combination of different neural network primitives (Convolutional Layers, Recurrent Layers, Mixture of Expert layers, etc).

An ultimate ambition of this research direction and many others in DL is to build extremely large heterogeneous networks that can solve hundreds or thousands of tasks, and deploy them in realistic use-cases. Computation, memory, and network bandwidth become extremely costly at this scale. To remedy some of these concerns, there has been prior research introducing sparsity to very large, deep networks through conditional activation.

The other key problem posed by very large, sparse, multitask networks (LSMNs) is managing the complexity they present at scale. Modern state of the art deep networks, especially those that tackle complex multitask settings, are built using tensor level processing abstractions provided by programming libraries and frameworks. However, implementing, scaling, and ultimately deploying LSMNs described above at this level of abstraction presents a workload that is almost intractable through a manual engineering process. Using higher level abstractions provided by other frameworks designed for rapid implementation of simpler networks is also an issue, as it limits the capacity for building highly custom modules, which are key to building high parameter, sparse networks that tackle tasks across various modalities.

This set of research notes explores various topics and ideas that may prove relevant to large, sparse, multitask networks and explores the potential for a general approach to building and managing these networks. A framework to automatically build, update, and interpret modular LSMNs is presented in the context of current tooling and theory. These notes are by no means comprehensive, and are meant to serve as the preliminary foray of a much deeper analysis of LSMNs and the ideas presented here.

These notes are structured so as to present a foundation of several ideas that may prove useful to building LSMNs, followed by their synthesis into an overall framework. Each preliminary section is meant to serve as a high level overview of the key ideas of the topic as relevant to LSMNs, rather than a comprehensive survey.

The document is organized into roughly 2 sections. The first covers some high level foundations to several of the key ideas mentioned in the second section, which discusses the motivation and approach to building LSMNs.

Topic List for Part 1:

1. Key ideas around multitask and multimodal learning and the role of computational primitives in deep neural networks is discussed.

2. Sparsity and conditional computation.

3. Composition of neural network modules in the context of both theory and modern engineering practice.

4. The automation of network design is discussed as a key step in building and managing large, often heterogeneous networks.

5. The Lottery Ticket Hypothesis, and model pruning is briefly also touched on.

Topic List for Part 2:

6. A preliminary general approach to building unified LSMNs is introduced. In this approach, several of the ideas above are synthesized, and some early ideas around the abstractions and processes required to radically simplify the construction, management, and understanding of LSMNs are put forward.

7. Further directions and vision, including a tool that encapsulates this approach and relation to the concept of software 2.0, is also mentioned.

2 Multitask and Multimodal Learning

Much of the work done in Machine Learning has been focused on optimizing a single objective or learning task. Multitask learning instead focuses on building models that solve usually related tasks and build a shared representation. The motivation behind Multitask learning is several fold, including obvious parallels to the biologically inspired learning, but the main motivation from a machine learning perspective is that introducing more than one learning task to a single model introduces a sort of useful inductive transfer, leading to models that may generalize better. There has been significant work in multitask learning including a variety of architectures, parameter sharing schemes, auxilary task introduction, and other methods. NLP tasks have greatly benefited from multitask learning, and it has achieved success in machine translation and speech recognition models as well. For a thorough treatment on multitask learning, several works are useful, including [1] and [2]

Of key interest in the research direction being explored here is the notion of Multimodal learning. The objective for this line of thinking can be surmised as building a unified deep architecture that can solve tasks across these various domains, be it vision, text, or otherwise. In this case, and in the general literature, domains are referred to as modalities. A key work on this topic is [3], where a single architecture, referred to as the Multimodel, was constructed to perform well on 8 different data corpora, including Imagenet, WSJ speech, COCO image captioning, English-German translation, and English to French translation. This work introduces several interesting ideas that are commonly seen in some other areas of DL research, including the composition of sub networks, the use of different primitives for different functons, and the introduction of a mixture of experts layer for conditional computation. Accordingly, it is useful to examine the details and architecture presented in this paper in some more detail, for the sake of following discussions. There are other multimodal architectures, including [4].

There are 3 key ideas introduced in the architecture outlined in [3] that are critical to the general idea behind LSMNs. The first of these is the idea of small modality specific subnetworks that convert input data into a unified representation within the model and vice versa. The second is the idea that different ”computational blocks” are useful for different problems. The third is an Encoder-Decoder architecture similar to other work [5] which is outlined in later sections.

2.1 Modality Nets

The authors introduce modality nets, which are essentially subnetworks that transform the various data inputs a joint latent representation. These networks are named like this because a specific subnetwork is designed for a specific modality (speech, text, images, etc). The authors specifically design modality nets to be computationally minimal. There is also an autoregressive design constraint enforced on these subnetworks, though that is outside the general purview of this set of notes. Modality nets are very similar to the datatype networks introduced in [5], and the neural module networks introduced in [6]. They can be viewed as a specific case of the general notion of neural module blocks introduced in this set of notes.

2.2 Computational Blocks

The authors also make note that the architecture makes use of various kinds of ”computational blocks” or primitives and that this is critical to achieve good performance on various problems. This makes intuitive sense. Examples of

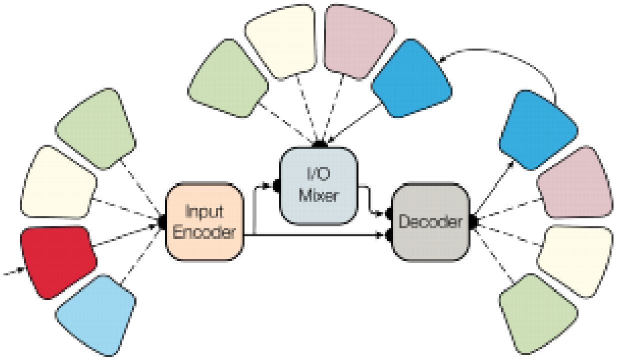

Figure 1: MultiModel Architecture Overview, adapted from [3]. Modality nets are colored, textless blocks

computational blocks are separable convolutions [7], attention mechanisms [8], sparsely gated mixture of experts [9] (to be explored in the next section on Sparsity), and others. The idea of fundamental building blocks that correspond to specific functionality is also critical to the notion of neural module blocks introduced in this work.

2.3 Encoder Decoder Architecture

The other key idea this work provides that will prove useful to us is the introduction of a key architectural paradigm used in DL literature, and in multitask learning in particular. This is the idea of an Encoder-Decoder pair. In [3], the Multimodel uses an Encoder, Mixer, Decoder architecture, similar to those found in sequence to sequence models [10]. This architecture is important, as it is a specific case of the general ECD architectural pattern introduced by [5], something analyzed in significant detail in later sections.

Key Takeaway: Key ideas are introduced in building neural networks for multimodal tasks: 1. Modality nets are small subnetworks that are composed together in the general architecture with the goal of mapping different input modalities to a unified latent representation. 2. Different computational blocks (CNN layers, mixture of experts, etc) are key to good performance on various tasks. 3. An Encoder Decoder architecture is introduced, a specific case of a powerful architectural pattern demonstrated by the approach outlined in the Ludwig paper outlined in later sections.

3 Sparsity through Conditional Computation

Key to building extremely large, multimodal networks is the idea of sparsity. Here, sparse activation is defined by conditional computation paradigms that have been introduced in various works. Extremely large neural networks are limited in their capacity because of the sheer number of parameters. Conditional Computation promises a reduction in computation despite increases in capacity.

Significant research into conditional computation through gating decision processes has been done [11] [12] [13]. These decisions could be discrete or continuous, and may also be stochastic. The Mixture of Experts layer used in the MultiModel was originally used in a very large, sparse recurrent language model introduced in [9]. The authors introduce a new layer/module that consists of a variety of small feed forward networks that act as experts, and a trainable gating network serves to select a sparse combination of the experts to process inputs. If we let $G(x)$ denote our gating network, and $E_1....E_n$ represent a set of n expert networks in a give mixture of experts layer, then the output y of the MoE module can be written as:

$$y = \sum_{i=1}^{n} G(x)_i E_i(x) \tag{1}$$

where $E_i(x)$ is the output of the i-th gated network. Gating functions can range from softmax gating to a noisy selection of the top k candidates, as described in [9]. Training of the gating network is done jointly with the rest of the architecture through simple back-propagation.

This sort of gating is key to building large, sparse networks that handle many multimodal tasks, and can be applied at both the specific layer level of abstraction as outlined above, or at the whole architecture level. Some form of sparsity is key to increasing model capacity using a reasonable amount of compute.

4 Neural Network Composition and Modular Architectures

A large interest in representation learning is understanding how learned representations are structured. Manually designed models often have come degree of compositionality. In [14], compositionality is broadly defined as combining simple parts to represent complex concepts. Various approaches to composing submodules, like the modality nets mentioned in [3]. In this section, we'll cover neural module networks [6], a useful line of research that demonstrates compositionality, and relate the Encoder-Decoder architectural paradigm mentioned earlier to the general concept in the context of Ludwig [5], a popular declarative deep learning tool.

5 Automating Neural Network Design

AutoML [15] is a popular topic in the field over the past few years, and broadly refers to any class of methods that automatically find and fit a machine learning method to a particular problem. In this section, we're specifically going to examine Neural Architecture Search [16], a group of strategies to automatically design deep learning architectures for given training objective. NAS methods have been applied to multitask and continual learning settings. [17]

6 Pruning Neural Networks

Pruning neural networks refers to selectively removing non contributing parameters while maintaining overall performance in a neural network. Pruning can be useful because of computational limitations. The Lottery Ticket pruning method [18] outlines an approach that finds a lottery ticket subnetwork within a larger deep neural network. This subnetwork may have as little as a tenth of the parameters, but it reaches the same performance the original network does, and sometimes generalizes to out of set examples better. For our purposes, pruning provides a useful scheme to treat deep networks as primitives to larger more complex networks by reducing their structure. This idea is explored further in the approach to building LSMNs outlined later.

7 An approach to building LSMNs

Building dynamically updated learning systems that can perform across thousands of tasks has been one of the great goals of the field, and will likely have enormous impact across many different application verticals. Such systems provide an approach towards complex problems that have historically been out of reach for learning systems. Approaches using end to end multitask learning systems have already been applied to notoriously difficult problem domains like autonomous navigation and medical applications.

7.1 Issues and Requirements

However, building and managing LSMNs provides a host of problems, including managing the complexity these networks at scale and in production. LSMNs by design are heterogeneous, large, and have many parts that will require constant change and management to learn new tasks and improve current ones. They involve the design and use of a variety of neural network primitives, and human interpretable organization to intervene in the design and training of LSMNs. Most popular frameworks and libraries that are used to build custom deep neural networks involve abstractions at the tensor level, providing useful algebra primitives and automatic differentiation. While this has saved researchers and engineers considerable time and been partially responsible for the speed with which neural network research has progressed, it does not provide a useful abstraction over the complexity required to implement LSMNs.

In light of these requirements, an approach to building large scale LSMNs would involve 2 key design decisions:

1. A general abstraction paradigm that encapsulates subnetworks of the pattern described in [3] [5] [14] [6] and others.

2. An automated method for both micro level primitive design and macro level primitive composition, preferably while maintaining intervenability.

7.2 Unifying Key Ideas

We can unify the notion of modality networks presented in [3], the MOE layer in [9], and neural module networks [6] along with the higher type based abstractions and encoder-decoder blocks presented in [5] to a general idea of modular neural network primitives associated with an information processing function. These primitives can range in function from simple tensor algebra primitives for functions like convolutions to complex architectures like Residual Networks. This level of abstraction is intentionally loose, as it leaves the internals of modules flexible, and allows for organization through functional definitions as opposed to mechanistic ones. Practically, this allows researchers and engineers to recycle blocks associated with large scale processing with necessary, but also include custom blocks for new information transforms in novel usecases. Key to this idea is the ability to generate new primitives on the fly, something to be discussed in detail. In organizing through function, the resulting modules maintain a high level of interpretability, key to productive engineering practice, even in exceedingly complex networks that involve heterogenous information processing and conditional computation across a sizeable number of parameters. Organizing through function provides the data type flexibility provided in [5] while allowing primitive definition to be more specific and with smaller scope if necessary.

While this general descriptor of neural network modules that can be extended from tensor level primitives to full blown SAO deep learning architectures is useful, in and of itself it does not increase the efficiency of building LSMNs, only of understanding and interpreting them. To rapidly prototype and build LSMNs as a collection of neural network modules with conditional computation, an automated method is necessary. While Neural Architecture Search, explored earlier, has mostly been used in the design of large, fully activated deep learning architectures, the framework provides a useful paradigm for the dynamic design of both individual layers or cell types, as well as higher level network topologies [19] [15]. We can differentiate between the two types of search based on their domains: Micro level searches generate new primitive types, while Macro level searches generate topologies composed of conditional routing functions and neural network modules. While not explicitly clarified before, the benefit of micro level search is that it aids in the discovery of new functional primitives for neural network when combined with a pruning step like the method outlined in [18]. This is significant, as it allows researchers to dynamically create new primitives, or use predefined ones. Rule based approaches, like those outlined in the Combiner approach from the Ludwig architecture [5] also provides a useful tool for the composition of neural network modules into a larger network architecture.

7.3 A Conceptual Approach to building LSMNs

Now, through the two synthesized ideas presented above, a general approach to the design and management of LSMNs begins to emerge. LSMNs can be composed of neural modules, a generic abstraction that organizes encapsulated functionality at defineable granularity.

Building NN Modules: NN modules can be predefined, using well known architectures or tensor level operations, ranging from a convolutional or recurrent primitive to a full blown ResNet. To define the best primitive for a new task, a micro level neural architecture search can be used, followed by a pruning operation to reduce the network to minimal parameter size. In practice, this makes is significantly easier to reuse, extend, and maintain good engineering practices around building very large networks.

Topology Generation: To generate the LSMN topology, a macro neural architecture search process can be defined to operate on neural modules and routing functions. Since neural architecture searches are essentially independent processes of the actual task, search spaces can be user defined and block architectures can be put together. Pretrained best weights can be swapped in from neural network modules as necessary, reducing overall training cost. Module interfaces can be modified by a separate process, similar to the Combiner introduced in [5]. Since modules are reusable, topology generation can be user defined instead of automatic, yielding networks similar to most cutting edge modular architectures. Whole network training could be done, or pretrained modules could be used out of the box.

Limited Architectural Opinions and Human in the Loop Learning: The strengths of this approach is that it doesn't force architectural notions on the user other than compositionality. A core concern raised when discussing automated neural network methods is that they search combinatorial spaces of the primitive types fed to them, and that human defined search spaces are providing more innovation than the search method entirely. The approach outlined here sidesteps this issue, by allowing module definition to be generic, sourced through a search or through manual design processes. This means that as new computational primitives are discovered and formalized, they can be immediately used in this approach. This could be layer types, whole networks, and routing function. While a tool will be proposed for deep learning, this generic composition + search method could conceivably used on other network types, including spiking neural networks. If you can define a computational primitive and it can be composed, it can be a part of the network. This could most certainly include non neural network computation as well, as any process would be entirely encapsulated and learning could be done in a one shot or independent fashion if necessary. The level of interpretability and extensibility, and the fact that automation is not a requirement demonstrates the ability of human in the loop learning with this approach.

The approach outlined above, while being framed as being useful to the building of extremely large, sparse, multiobjective networks, also provides a generic framework in which to design deep architectures for one task. Encapsulation and automation are useful features for any network generation process, and are already used to some degree in the construction of many networks, including ResNets and seq to seq models.

There are several areas still being developed: scoped out research ideas, extensions and also a direction to implement a set of tools that would allow researchers and engineers to build heterogeneous networks using this approach, and eventually build LSMNs.

References

[1] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, 2017.

[2] Y. Zhang and Q. Yang, "A survey on multi-task learning," *arXiv preprint arXiv:1707.08114*, 2017.

[3] L. Kaiser, A. N. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit, "One model to learn them all," *arXiv preprint arXiv:1706.05137*, 2017.

[4] S. Pramanik, P. Agrawal, and A. Hussain, "Omninet: A unified architecture for multi-modal multi-task learning," *arXiv preprint arXiv:1907.07804*, 2019.

[5] P. Molino, Y. Dudin, and S. S. Miryala, "Ludwig: a type-based declarative deep learning toolbox," *arXiv preprint arXiv:1909.07930*, 2019.

[6] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, "Neural module networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 39–48.

[7] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[8] W. Wang and J. Shen, "Deep visual attention prediction," *IEEE Transactions on Image Processing*, vol. 27, no. 5, pp. 2368–2378, 2017.

[9] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *arXiv preprint arXiv:1701.06538*, 2017.

[10] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[11] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[12] E. Bengio, P.-L. Bacon, J. Pineau, and D. Precup, "Conditional computation in neural networks for faster models," *arXiv preprint arXiv:1511.06297*, 2015.

[13] K. Cho and Y. Bengio, "Exponentially increasing the capacity-to-computation ratio for conditional computation in deep learning," *arXiv preprint arXiv:1406.7362*, 2014.

[14] J. Andreas, "Measuring compositionality in representation learning," *arXiv preprint arXiv:1902.07181*, 2019.

[15] M.-A. Zöller and M. F. Huber, "Survey on automated machine learning," *arXiv preprint arXiv:1904.12054*, 2019.

[16] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *arXiv preprint arXiv:1808.05377*, 2018.

[17] R. Pasunuru and M. Bansal, "Continual and multi-task architecture search," *arXiv preprint arXiv:1906.05226*, 2019.

[18] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.

[19] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.