Understanding the Capabilities and Limitations of Neural Networks for Multi-task Learning

Vatsal Sharan[1] **Xin Wang**[2] **Brendan Juba**[3] **Rina Panigrahy**[2]

vsharan@stanford.edu, wanxin@google.com, bjuba@wustl.edu, rinap@google.com

[1]Stanford University, [2]Google, [3]Washington University in St Louis

Abstract

Can deep learning solve multiple, very different tasks simultaneously? We investigate this question in this work, and do a systematic study of how the properties of the underlying tasks affect the ability of a single neural network to learn them jointly. We present theoretical and empirical findings that neural networks are capable of learning multiple tasks when these tasks are naturally clustered and well-separated. We also show that multi-task learning can be provably hard—even though the individual tasks are easy to learn—in settings where such a natural separation does not exist, and validate this through experiments.

1 Introduction

A popular hypothesis in neuroscience is that the human brain is modular [1, 2, 3]. Though the presence of some level of modularity in the brain is accepted, there is significant debate about the extent of modularity [4]. It is more or less accepted that there are dedicated modules for low-level tasks such as vision and audio processing, but it is unclear and much debated whether modularity also exists for higher level reasoning and cognitive tasks.

In this work, we study this question of whether modularity is necessary for learning. We investigate this in the context of multi-task learning, where the goal is to learn a single model to solve multiple tasks. Humans have the ability to learn thousands of tasks in the course of their lifetime. If we want to train models with similar abilities then should such models necessarily have some modularity, such as modular components to individually solve the different tasks? Specifically, we investigate this in the context of neural networks, and ask the following question:

Can one giant neural network simultaneously solve multiple tasks?

There is significant interest in jointly solving multiple tasks using neural networks, and many successful cases of multi-task learning using neural networks are known (we discuss some of these in Appendix B.1). However, we are unaware of any prior systematic study on the ability of a single neural network to jointly solve multiple tasks. In particular, our goal is to understand multi-task learning in settings where the different tasks do not necessarily share any internal representations, and hence learning one task has limited or no direct utility for learning other tasks. It is unclear if a single neural network can learn such tasks, because training for these different tasks could possibly interfere with each other leading to either poor optimization or generalization. In this work, we methodologically evaluate if and when neural networks can solve multiple problems of this form, with both a theoretical and an empirical study.

1.1 Main Results and Overview

We now summarize our setup and results and give an overview of the paper. We show that a key property which determines learnability of multiple tasks is the degree of separation of the data points

NeurIPS 2019 Workshop on Machine Learning with Guarantees, Vancouver, Canada.

from the different tasks. To demonstrate this, we consider two different scenarios for multi-task learning and examine them theoretically and empirically.

1. *Data is non-trivial to separate into individual tasks (Section 2):* The first scenario we consider is when the data-points from the different tasks are non-trivial to separate out. This would be the case if the data distributions for the different tasks are close to each other.
2. *Data is naturally clustered into individual tasks (Section 3)*: The second scenario is when the data distributions for the different tasks are well-separated. In this case, doing any simple clustering (such as a k-means clustering) on the combined data would cluster the data based on the task.

Theorem 1 shows that learning multiple tasks using a single neural network can be provably hard in the first case—even though the individual tasks are easy to learn. Motivated by the theoretical result, in Section 2.2 we construct a synthetic setting which empirically demonstrates that learning multiple tasks with a single neural network can be hard when the different tasks are difficult to separate out. By systematically varying the degree of separation of the different tasks in the synthetic setting, we also show that making the different tasks easier to separate significantly helps a single neural network to learn them. We investigate this further in Section 3 and empirically show that when the data from the different tasks is well-separated into distinct clusters, a single neural network can simultaneously do well on all of them. Theorem 2 provides theoretical evidence for this ability of neural networks to learn multiple well-separated tasks, using the recent framework of Arora et al. [5].

2 Data is Non-trivial to Separate into Individual Tasks

We consider the case when the data for the different tasks is non-trivial to separate out. We first show that there exist certain settings in this case where a single learner provably needs a significantly higher time or sample complexity to learn multiple tasks, than what is required to solve each individual task. Based on the theoretical result, we construct a synthetic task where a single learner for multiple tasks needs a much larger number of samples than a separate learner for every task. We then relax the synthetic setting, and show that a single network does significantly better if the data from the different tasks becomes easier to separate out.

2.1 Theoretical analysis

We describe a setting where using a single network is provably bad. The data points are $b + d$ dimensional binary vectors. The label of any data point is the XOR ($\oplus$) of some unknown parity function computed on the first b bits and a simple function computed on the last d bits (any easy to learn function suffices, for e.g. a function which only depends on the last bit), along with some noise.

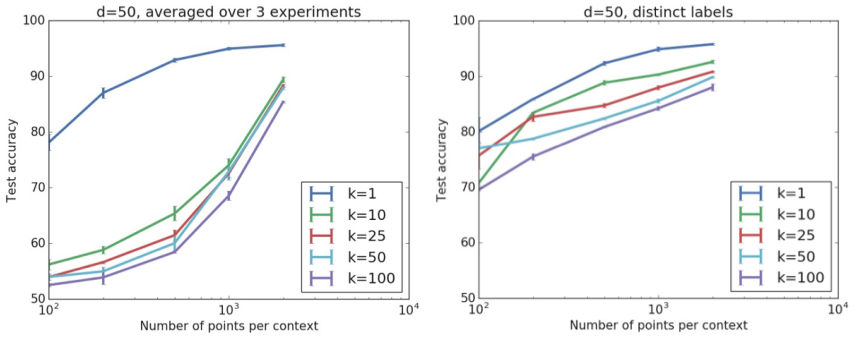
Formally, for any $b + d$ dimensional datapoint x, let $x[: b]$ denote the first b bits of x, and let $x[b :]$ denote the last d bits. Let $v \in \{0, 1\}^b$. Let e be a Bernoulli random variable which is 1 with probability 0.1. The labelling function f is a function from $\{0, 1\}^{b+d} \to \{0, 1\}$, defined as follows,

$$f(x) = ((\langle v, x[: b]\rangle \mod 2) \oplus e) \oplus (\text{any simple function computed on } x[b :]).$$

Note that learning f is also a multi-task learning problem where the individual tasks are easy. The reason for this is as follows. Suppose we have multiple tasks with a separate simple classification function for every task, with the function being determined by a parity computed on the first b bits. Then solving the multi-task learning problem is equivalent to learning f. Note that learning each individual task here is easy as it only requires learning the simple function.

We show hardness of learning f using any neural network whose architecture is symmetric to the coordinates of the input and hence is invariant to the ordering of the coordinates. We consider such symmetric architectures as they represent a vanilla, giant network instead of a more modular network. Note that almost all commonly used neural networks architectures treat all coordinates of the input symmetrically and hence Theorem 1 applies to them. The proof relies on the hardness of learning *parity with noise*, the definition of the problem and the proof overview appears in the appendix.

Theorem 1. *Conditioned on the hardness of the parity with noise problem, any model which treats all coordinates symmetrically—and hence is invariant to a permutation applied to the coordinates of all the data points—needs at least $\Omega(d^b)$ time or samples to learn f. Hence a vanilla neural network which is invariant to a permutation of the coordinates needs at least $\Omega(d^b)$ time or samples to learn f. In contrast, learning a separate model for every set of the first b bits needs $O(2^b)$ samples.*



(a) Test accuracy vs. number of points per context (task) for a binary classification problem.

(b) Test accuracy vs. number of points per context (task) when the labels for each task are different.

Figure 1: Results for experiments when there is a different linear classifier for each task, and the task index is appended as a prefix to each data point. We see that the performance of a single neural network drops significantly when the number of tasks k increases when the labels are binary and shared across tasks, but the drop is much less when each task has its own set of labels.

2.2 Experiments

Our experimental setup is similar to the problem described in the previous section. The setup is that the data points come from different tasks, with a different ground truth classifier for every task. We now describe the setup formally. The problem is binary classification and the input is $b + d$ dimensional. The first b bits denote a prefix which indicates the task from which that data point is coming. Depending on the task, a different linear classifier is applied to the last d coordinates of the data point, and the (binary) label is determined by the output of this linear classifier.

Fig. 1a shows the result on fitting a single neural network to this problem. We see that there is a significant gap between the accuracy when there is only one task ($k = 1$) and when there are multiple tasks. This shows that learning all tasks using one neural network is challenging in this case.

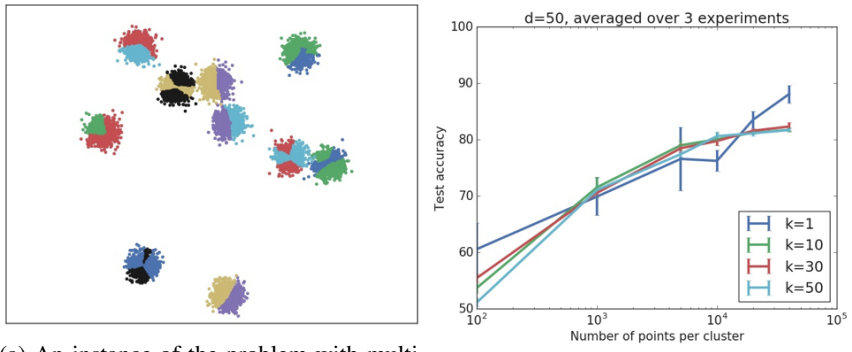
Making the tasks easier to separate: The previous problem is possibly hard for a single network because it has to learn that to apply a different linear classifier based on just a first few coordinates. In the following experiments, we make the data from the different tasks easier to separate out.

1. Instead of a binary classification problem with the same $\{0, 1\}$ label across all tasks, we have a separate set of labels for each task, i.e., $\{0, 1\}$ for the first task, $\{2, 3\}$ for the second task, and so on. Hence there are $2k$ labels in total for the k tasks.
2. Instead of a binary encoding of the task index, we have a one-hot encoding. Hence the prefix now has k bits, with only the bit corresponding to the task of that data point being one.

Fig. 1b shows the result of the first experiment where there are separate labels for every task, the result of the second experiment appears in Fig. 3 in the Appendix and is similar to Fig. 1b. We can see that there is a much smaller gap between the performance of training a separate model for every task (which corresponds to $k = 1$) and training a single model for multiple tasks (corresponding to higher values of k). Hence a single network does a much better job of learning multiple tasks than in the previous case when the tasks were harder to separate.

3 Data is Naturally Clustered into Individual Tasks

In the previous experiment, we saw that a single neural network gets a reasonable performance on multiple tasks if the tasks are easy to separate out. We explore this further in this section, and demonstrate through synthetic experiments and theoretical results that a single neural network can learn multiple tasks if the tasks are well-separated into clusters. We consider two cases, first when the labels within each cluster are determined by a simple linear classifier (Appendix B), and second when the labels are given by a teacher neural network.



(a) An instance of the problem with multiple clusters, and a teacher neural network to determines the labels for each cluster.

(b) Test accuracy vs. number of points per cluster

Figure 2: Experiment where data is clustered into tasks. A single neural network does well even when there are multiple clusters.

Teacher neural network for each task. Here the data is drawn from a mixture of k well-separated Gaussians. Within each Gaussian, the data points are marked with either of two labels, with the labels determined by a teacher neural network. A different teacher network is sampled at random for every cluster. The teacher network has one hidden layer, with 10 hidden units. Fig. 2a below shows an instance of this task in two dimensions. Fig. 2b shows the performance of a single neural network trained on this task. The performance of the neural network changes only slightly on increasing the number of clusters (k), suggesting that a single neural network can learn across all clusters.

3.1 Theoretical analysis

It is challenging to that that neural networks can provably learn multiple functions because we do not have a good understanding of the functions that neural networks can learn. However, we show that neural networks can provably learn multiple functions from a certain class, using a recent framework of Arora et al. [5].

Arora et al. [5] recently showed that certain classes of smooth functions can be learnt by gradient descent, despite over-parameterization. We show that if the data points corresponding to each task individually satisfy the property in Arora et al. [5], then the union of those data points also satisfies the property in Arora et al. [5] and hence can be learnt by a single neural network—as long as the data points from the different tasks are orthogonal.

Theorem 2. *If the data-points for different tasks are orthogonal and if the tasks individually satisfy the property in Arora et al. [5], then the overall problem satisfies the property in Arora et al. [5] and can be learnt by a single neural network with sample complexity equal to the sum of the sample complexity of learning the individual tasks.*

4 Conclusion

Our results, and other related literature, indicate that a single neural network could have the ability to learn tasks across multiple, diverse domains. However, having a modular architecture still has other benefits, such as the following:

1. *Energy consumption and computational efficiency:* Modular learners use less energy and computation, as only a subpart of the entire model needs to evaluate any data point. This also leads to smaller models which use less memory.
2. *Interpretability:* Modular architectures can be more interpretable, as it is clearer what role each part of the model is performing.

It would be very interesting if some of these benefits of modularity could be extended to a single model as well. For instance, is it necessary for a giant, non-modular network to have sub-parts which perform simple computations, and which can be extracted from the larger network? This could help in interpreting and understanding large neural networks.

References

[1] Jerry A Fodor. *The modularity of mind*. MIT press, 1983.

[2] David C Geary and Kelly J Huffman. Brain and cognitive evolution: Forms of modularity and functions of mind. *Psychological bulletin*, 128(5):667, 2002.

[3] David Meunier, Renaud Lambiotte, Alex Fornito, Karen Ersche, and Edward T Bullmore. Hierarchical modularity in human brain functional networks. *Frontiers in neuroinformatics*, 3: 37, 2009.

[4] Richard Samuels. Is the mind massively modular? 2006.

[5] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *arXiv preprint arXiv:1901.08584*, 2019.

[6] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

[7] Henry Tsai, Jason Riesa, Melvin Johnson, Naveen Arivazhagan, Xin Li, and Amelia Archer. Small and practical bert models for sequence labeling. *arXiv preprint arXiv:1909.00100*, 2019.

[8] Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351, 2017.

[9] Roee Aharoni, Melvin Johnson, and Orhan Firat. Massively multilingual neural machine translation. *arXiv preprint arXiv:1903.00089*, 2019.

[10] Ankur Bapna, Colin Andrew Cherry, Dmitry Dima Lepikhin, George Foster, Maxim Krikun, Melvin Johnson, Mia Chen, Naveen Ari, Orhan Firat, Wolfgang Macherey, et al. Massively multilingual neural machine translation in the wild: Findings and challenges. 2019.

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

[12] Lukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. One model to learn them all. *arXiv preprint arXiv:1706.05137*, 2017.

[13] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

A Theoretical Results

A.1 Hardness of Learning when it is Non-trivial to Separate Sub-tasks

Definition 1. *(parity with noise) Let v be a vector in $\{0,1\}^d$. Let e be a Bernoulli random variable which is 1 with probability 0.1. The parity with noise problem is defined as the problem of learning the function g from $\{0,1\}^d \to 0,1$, defined as follows,*

$$f(x) = ((\langle a, v\rangle \mod 2) \oplus e).$$

It is widely believed that learning noisy parities is hard, and variants of the problem are used as a cryptographic primitive. The best algorithm known for the problem is essentially only as good as a brute force search. In particular, the following hardness result is conjectured.

Conjecture 1. *(parity with noise) Any algorithm for learning noisy parities needs roughly*[1] 2^d *time or samples. Also, even if the unknown vector v is known to be b-sparse, any learning algorithm still needs roughly d^b time or samples.*

[1] It is known that logarithmic factors can be shaved from the exponent.

Based on the hardness of learning parities, we show the following hardness result for learning f.

Theorem 1. *Conditioned on the hardness of the parity with noise problem, any model which treats all coordinates symmetrically—and hence is invariant to a permutation applied to the coordinates of all the data points—needs at least $\Omega(d^b)$ time or samples to learn f. Hence a vanilla neural network which is invariant to a permutation of the coordinates needs at least $\Omega(d^b)$ time or samples to learn f. In contrast, learning a separate model for every set of the first b bits needs $O(2^b)$ samples.*

Proof sketch: The main idea behind the proof is that if the model is symmetric to a permutation being applied to all the coordinates, then it needs to solve a sparse parity with noise problem over $d + b$ coordinates where the unknown vector is known to be b-sparse. The reason for this is because we could apply a random permutation to the coordinates of all data points and give this as the input to the model, and the time and sample complexity of the model would be invariant to the change. Applying a random permutation to the coordinates converts the original parity with noise problem where the unknown vector was only supported on the first b coordinates to a general sparse parity with noise problem, where the unknown vector is b-sparse. Solving noisy parities where the unknown vector is known to be b-sparse over $d + b$ coordinates requires at least d^b time or samples, and hence the result follows. □

A.2 Learnability when Data is Naturally Clustered into Separate Sub-tasks

Theorem 2. *If the data-points for different tasks are orthogonal and if the tasks individually satisfy the property in Arora et al. [5], then the overall problem satisfies the property in Arora et al. [5] and can be learnt by a single neural network with sample complexity equal to the sum of the sample complexity of learning the individual tasks.*

Proof sketch: Arora et al. [5] show that if the following Gram matrix corresponding to the ReLU activation function is well-behaved, then the function is learnable with gradient descent. Let $\{x_i \in [n]\}$ be the training set. The matrix $H^\infty \in \mathbb{R}^{n \times n}$ is defined as follows.

$$H^\infty_{ij} = \frac{x_i^T x_j(\pi - \arccos(x_i^T x_j))}{2\pi}, \ \forall \ i, j \in [n].$$

Informally, Du et al. [6] show that the gradient descent on a two layer ReLU network converges to small error as long as the minimum eigenvalue of the matrix H^∞ is bounded from below. Arora et al. [5] additionally bound the generalization error, and show that for any 1-Lipchitz loss function the generalization error of the two layer ReLU network found by gradient descent is at most

$$\sqrt{\frac{y^T (H^\infty)^{-1} y}{n}}.$$

Here y is the vector of the labels. Therefore, if $y^T (H^\infty)^{-1} y$ is a constant for a task, then the generalization error decreases at a $\sqrt{1/n}$ rate.

Note that if the data points for the individual tasks are orthogonal, then the cross-terms in the matrix H^∞ are 0 and hence $y^T (H^\infty)^{-1} y$ for the overall problem is just the sum of the contributions from each of the tasks. Hence the overall problem can be learned with sample complexity equal to the sum of the sample complexity of learning each individual task.

If the data points for the tasks are close to being orthogonal, then we can use matrix perturbation bounds to control the error and show that the property is still approximately satisfied (currently ongoing work). □

B Additional Experiments for Section 3

Linear classifier for each task: In this experiment, the data is drawn from k clusters, and from a mixture of two well-separated Gaussians in each cluster. Data points from the two Gaussians within each cluster are assigned two different labels, for $2k$ labels in total. Fig. 4a below shows an instance of this task in two dimensions, the red circles represent the clusters, and there are two classes

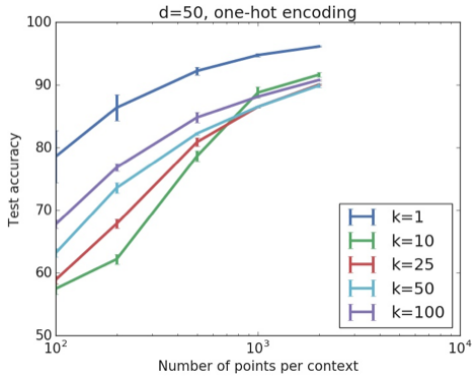
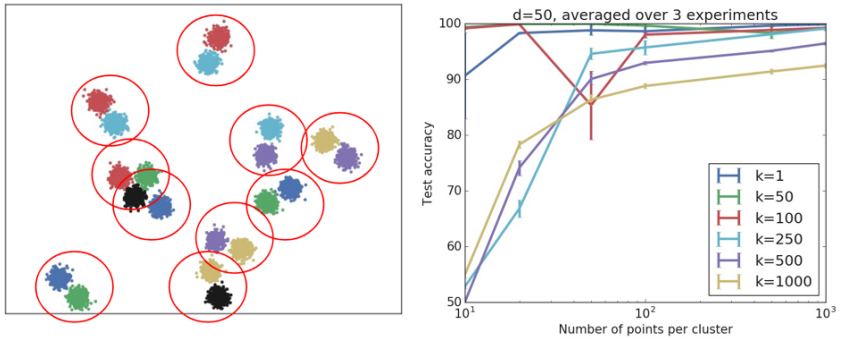


Figure 3: Test accuracy vs. number of points per context (task) when the task label is encoded in one-hot representation and hence the tasks are easier to separate.



(a) An instance of the problem with multiple clusters, each cluster is indicated by a red circle.

(b) Test accuracy vs. number of points per cluster

Figure 4: Experiment where data is clustered into tasks. A single neural network does well even when there are multiple clusters.

drawn from well-separated Gaussians from each cluster. In high dimensions, the clusters are very well-separated, and doing a k-means clustering to identify the k cluster centers and then learning a simple linear classifier within each cluster gets near perfect classification accuracy. Fig. 4b shows the performance of a single neural network trained on this task. We can see that a single neural network still gets good performance with a modest increase in the required number of samples.

B.1 Related applied work

The success of neural networks at multi-task learning has also been demonstrated in more applied work. Here, we outline a few examples:

1. In natural language processing, Tsai et al. [7] show that a single model can solve machine translation across more than 50 languages. As reflected in many other works as well [8, 9, 10, 11], a trend in NLP seems to be to use one model for multiple languages, and even multiple tasks.
2. Success has also been achieved in training a multiple model across tasks in very different domains, such as speech and language [12].
3. There is also work on training extremely large neural networks which have the capacity to learn multiple tasks [13].