Interpretability in Gated Modular Neural Networks

Anonymous Author(s)

Abstract

Monolithic deep learning models are typically not interpretable, and not easily transferable. They also require large amounts of data for training the millions of parameters. Alternatively, modular neural networks (MNN) have been demonstrated to solve these very issues of monolithic neural networks. However, to date, research in MNN architectures has concentrated on their performance and not on their interpretability. In this paper we would like to address this gap in research in modular neural architectures, specifically in the gated modular neural architecture (GMNN). Intuitively, GMNN should inherently be more interpretable since the gate can learn insightful problem decomposition, individual modules can learn simpler functions appropriate to the decomposition and errors can be attributed either to gating or to individual modules thereby providing either a gate level or module level diagnosis. Wouldn't that be nice? But is this really the case? In this paper we empirically analyze what each module and gate in a GMNN learns and show that GMNNs can indeed be interpretable, but current GMNN architectures and training methods do not necessarily guarantee an interpretable and transferable task decomposition.

1 Introduction

There is skepticism about the reliability, fairness, explainability and interpretability of deep learned models. In her book, *Weapons of Math Destruction* [15], author Cathy O'Neil explains the disastrous and unfair fallout of using black box models which provide no insight into their decisions. Monolithic deep neural architectures create black box models with less interpretable results and do not facilitate transfer of knowledge.

This naturally led me to look into modular neural network (MNN) architectures. There are different modular neural network architectures [10, 19, 3]. Gated modular neural networks (GMNN) are currently the most successful MNNs. They consist of a set of simple individual neural network modules (no shared weights) that are trained along with another simple neural network that works as a gate or soft-switch. The gate determines which module should be used for each data sample. There are various methods for combining and training the modules and the gate. Usually the gate and the modules are trained on the same input data samples but this is not necessary.

Research in this area dates back at least 2 decades and has demonstrated the potential advantages of a GMNNs: (1) learn subtasks with much smaller networks and hence parameters [18]; (2) requires less data to train the smaller networks; (3) learnt subtasks could be applied across different complex tasks enabling transfer learning [14]; (4) possibility of easier parallelization and distributed computation [17, 18]; (5) leverage domain knowledge [16].

However, the research so far has concentrated on the overall performance of the GMNN and not on its potential to be interpretable. GMNNs can be inherently more interpretable since: (1) the modules and gate are simpler networks that can be debugged and explained more easily than a monolithic network; (2) the gate decomposition of the task between the modules could facilitate data driven error attribution to either the gate or the modules and the modules would be better suited for transferability. But is this really the case? None of the earlier work in GMNNs clearly addresses this

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

question by delving into what the individual modules learn and how the problems are decomposed among the modules by the different architectures and methods of training, a detail that is essential for understanding the interpretability of GMNNs. In this paper we attempt to bridge this gap in GMNN research.

Explainability and interpretability in deep learning is an active area of research [20]. There is still little consensus on what interpretability is and how to evaluate it for benchmarking [4, 12]. In our work we define interpretability as the data driven ability to attribute errors for debugging. With GMNN this could ideally be achieved by: (1) the gating network learning a meaningful decomposition of the input space into regions with natural 'rules'. For example, for a classification task the gate would use different experts to predict different classes; or (2) each expert learns non-intersecting functions or subsets of the task which implicitly satisfies case (1). Such a task decomposition would enable error attribution either to gating or to individual modules based on the input samples.

In this paper we use the Mixture of Experts (MoE) architecture, which is the most commonly used GMNN architecture, to perform in-depth empirical analysis of what the modules and gate learn to address the following questions that can provide an insight into the potential of interpretability and transferabilty of the GMNNs:

1. How are tasks divided into sub tasks by the gate in the MoE architecture?
2. Does the gate decomposition of the task among the experts guarantee interpretability?
3. Does an interpretable task decomposition actually exist and can the gate learn it?
4. What are the pathological cases of task decomposition?

The main contributions of our paper are: (1) a detailed empirical analysis of what the experts and gate in MoE architectures learn. To the best of our knowledge, there is no such prior work; and (2) findings that (a) MoE architectures can indeed be interpretable; (b) however, not all MoE architectures are inherently interpretable; and (c) the existing MoE architectures and their methods of training cannot guarantee a problem decomposition that is either interpretable or transferable and hence, further work is required in this area. The code for all the experiments in this paper is at: `https://github.com/neuripsxai2021/neurips_xai_workshop_2021`.

2 Related Work

Interpretable and explainable deep learning is an active area of research. Xie et. al. [20] have recently nicely summarized the state-of-the-art in explainable deep learning research. They review different topics of explainability, such as, understanding the learning mechanism, model debugging, adversarial attack and defense, and fairness and bias in deep neural networks. Their survey shows that there is no significant work leveraging the inherent interpretability in modular neural networks.

In our work we are most interested in data driven error attribution for model debugging, that is, attributing the mislabeled errors to the part of the model that needs to be improved. This is more natural in modular neural networks where you can attribute the errors to either the gate or individual modules that can be further improved. Existing model debugging methods, such as: *ModelTracker* [2] provide a model agnostic tool to debug errors like mislabeled data or inadequate features, Alain et. al. [1] build linear classifiers, from the representations of the intermediate layers, to assess the quality of the representations to predict the labels, and *neural stethoscopes* [5] which analyses the deep neural network learning and the factors that influence it by promoting or suppressing information. It uses a 2 layer perceptron branch to do this. Hence, we see that existing methods use additional tools and mechanisms for debugging while modular networks could inherently provide this feature.

To the best of our knowledege there is no existing work that delves into the potential for interpretability in GMNNs. [9, 16] mention briefly that the MoE architecture can be interpretable but do not provide substantial evidence for it.

3 Mixture of Experts Overview

Mixture of Experts (MoE) is the simplest of the wider class of GMNN architectures, introduced by [7, 6], where data-flow is dynamically configured according to the input. Different MoE architectures

can be realized by different combinations of expert outputs mediated by the gate with different loss functions. Four of these architectures are presented here.

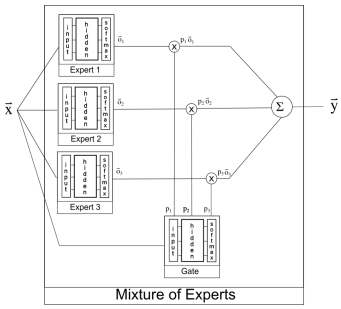


Figure 1: Mixture of experts architecture [6]

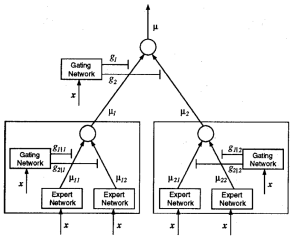


Figure 2: A two-level hierarchical mixture of experts [8]

3.1 Expectation Model

Jacobs et. al. [7] introduced the architecture in Figure 1 where the expert networks compete to learn the training patterns, and the gating network mediates this competition. After training, expert networks 1, 2 and 3 compute different functions that are useful in different regions of the input space. Let the vectors $\vec{o}_1, \vec{o}_2$ and $\vec{o}_3$ denote the outputs of the three expert networks. The gating network decides whether expert 1, 2 or 3 is currently applicable. Scalars p_l, p_2 and p_3 denote the 3 output units of the gating network. In general, the architecture may contain any number of expert networks. If there are n expert networks, then the gating network has n output units. The output of the entire architecture, $\vec{y}$, is the expected sum of the outputs of the individual experts, $\vec{y} = \sum_{i=1}^{n} p_i \vec{o}_i$, and the loss L is $L = [d - \sum_{i=1}^{n} p_i o_i]^2$, where d is the desired output. Since the output is a sum of proportions of the outputs of the experts, the experts are tightly coupled. Change in weights of one expert changes the residual error and hence affects the weights of all other experts.

3.2 Pre-softmax Model

I designed another architecture which is the same as Figure 1 but where the gate smoothly combines the outputs of each expert network before applying the softmax and then applying the softmax to the combined output, $\vec{y} = softmax(\sum_{i=1}^{n} p_i o_i)$. The loss L in this case is the negative log loss $L = -1/N \sum_{i=1}^{N} d_i \log(p(\vec{y} = d_i))$.

3.3 Stochastic Model

In their subsequent work, Jacobs et. al. [6], introduced a gate network that makes a stochastic decision of which expert output should be selected. The output of the entire architecture, $\vec{y}$ is therefore one of $\vec{o}_1$ to $\vec{o}_n$ expert outputs sampled according to the distribution learnt by the gate network. The expected loss L is then the expected sum of the loss of each expert, $L = \sum_{i=1}^{n} p_i [d - \vec{o}_i]^2$. Notice that in this loss function, each expert is required to produce the whole of the output vector rather than a residual. As a result, the goal of a local expert on a given training case is not directly affected by the weights within other local experts.

3.4 Hierarchical and Multilevel Models

Some problems require multi-level architectures. Jordan et. al.[8] proposed a hierarchical mixture of experts, shown in Figure 2, in which the experts and gate are generalized linear models. Learning is treated as a maximum likelihood problem and the parameters of the architecture are adjusted using Expectation-Maximization (EM) algorithm.

Another composition of multi-level architecture of experts and gates was proposed by Kirsch et. al. [9], shown in Figure 3. They too propose an EM algorithm to learn the parameters of the architecture. The difference between [8] and [9] is that while the former has experts only in the first level, the latter

is a composition of experts at each level where the choice of the experts at each level is determined by the gate at that level.

(a) Based on the input, the controller selects K modules from a set ofM available modules. In this example, K = 3 and M = 6.

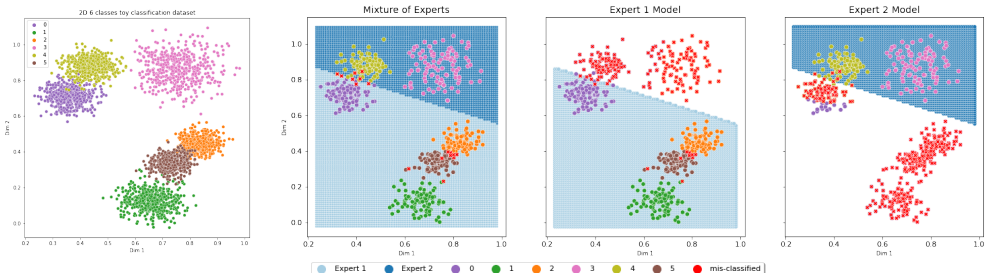
(b) The selected modules then each process the in- put, with the results being summed up or concatenated to form the final output of the modular layer.

Figure 3: Architecture of the modular layer [9]. Continuous arrows represent data flow, while dotted arrows represent flow of modules.

3.5 How are tasks divided into sub tasks by the gate in the MoE architecture?

It is important to see how the task is allocated to the experts by the gate and what the experts and the gate are learning in order to analyze interpretability. So, we implemented the different MoE architectures outlined in Section 3[1] with simple linear expert and gate networks. Since it is always best to start simple, we started with a toy classification problem of 2D mixture of Gaussians with 6 components, shown in Figure 4a. Each component corresponds to a class and so we have a 6 class classification problem. There are 2400 training samples and 600 test samples. We used RMSProp optimizer for updating the parameters. The experts and the gate are jointly trained on the loss for the corresponding architecture.

The inputs to all the expert and gate networks are the same. Multiple runs of the experiments with the four different models have shown that the expectation model consistently partitions the data interpretably for the toy classification dataset, whereas the EM, pre-softmax and stochastic models do not. So, in the interest of space we report the results of the expectation model for all our experiments in this paper. The results for multiple runs of all the 4 architectures for the toy dataset are in Appendix A.1. The expectation model is also the model that potentially allows reduction of computation, because the gate can select the best expert to use, and then use only that expert. The EM model has the slowest training time compared to the other 3 architectures.



(a) (b)

Figure 4: (a) Toy classification dataset with simple 2D 6 classes Gaussian mixture; (b) Test data class predictions by the trained MoE expectation model and the individual experts along with the samples gated to the expert by the gate (shaded regions). Red ‘x’ are the mis-classified samples.

[1] we did not implement the multi-level experts and gates but used the EM algorithm from Kirsch et. al. [9] to train a single level of experts and one gate to compare it to training with the other architectures that use back propagation.

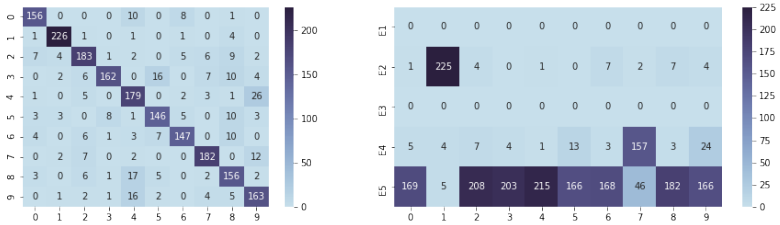
Figure 4b shows class predictions of the final learnt expectation model and each of the individual experts with 2 experts. We can see that the final model performs well on the toy classification task with few mis-classfications, marked as red 'x'. Figure 4b also shows the gate selection of the experts for samples of each class as the shaded regions in the final model. The class predictions by each expert in Figure 4b shows that the classification task of 6 classes is divided among the experts. That is, each expert learns to classify a subset of the classes as expected. The shaded region in each expert shows the samples gated to that expert by the gate during prediction.

4 Does the gate decomposition of the task among the experts guarantee interpretability?

An interpretable task decomposition, as mentioned earlier, is one in which either: (1) the gating network learns a meaningful decomposition of the input space into regions with natural 'rules'. For example, for a classification task it would use different experts to predict different classes; or (2) each expert learns non-intersecting functions or subsets of the task which implicitly satisfies case (1). Let us see if either or both these cases are achieved by the trained MoE models for the toy classification problem.

The gate partition of the input space, of the toy classification dataset, among the experts is shown in Figure 4b for the expectation model with 2 experts. We see that the model's gate partitions the input space linearly such that the samples of the same class are classified by the same expert. Hence we see that case (1) is achieved with simple linear experts and gate.

Now let us check for case (2). Figure 4b shows that there is a clean decomposition of the subtasks between the experts, satisfying case (2). That is, the subsets of classes learnt by each expert are non-intersecting. So, for the toy classification task, the expectation model with 2 experts achieves both cases (1) and (2). Hence, we see that GMNNs can produce interpretable task decomposition.



(a) (b)

Figure 5: (a) Confusion matrix of predictions of the MNIST test data by MoE expectation model (b) Experts used by the gate for classification of each digit

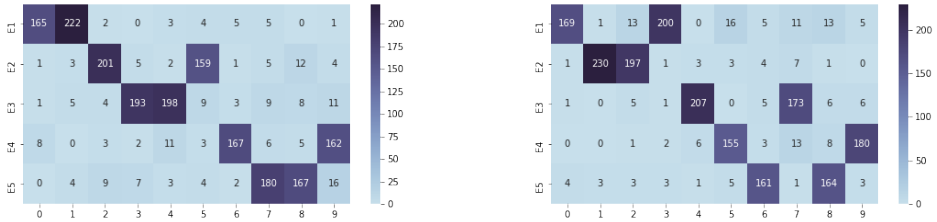
Let us proceed to the more complex problem of MNIST [11] digits classification and again check if the cases are satisfied. I used $\approx 10,000$ training samples and 2000 test samples containing all the digits. Each expert and the gate is a simple convolutional network with a single convolutional layer and 2 hidden layers with ReLU activation. From Figure 5a, the confusion matrix of the predictions by the MoE expectation model, we see that the model accuracy is good. However, from the expert utilization table in Figure 5b, we see that only 3 of 5 experts are used with experts 5 used for most of the samples. For some digits, such as, 7 and 9, both experts 4 and 5 are selected. So, for the more complex problem, the gate task decomposition is not interpretable as it does not achieve either case (1) or (2). Hence, we see that interpretability in GMNNs is not always guaranteed. Let us analyse the possible causes for this.

5 Does an interpretable task decomposition actually exist and can the gate learn it?

The simplest way to train a GMNN is to train the gate and experts at the same time, 'end to end', by gradient descent. During training the gating probabilities for each example determines which experts

get trained on that example, that is, gating interacts with training and in effect experts are trained only when they are chosen by the gating network. However, in section 3.5 we saw that the existing MoE architectures trained 'end to end' do not necessarily decompose the task interpretably especially for the more complex MNIST dataset. Why is this? Three possible hypothesis are that: (a) there is a training disadvantage for the interpretable task decomposition, that is, the training converges slower; (b) interpretable decomposition leads to a higher overall error rate; or (c) the heuristic of jointly optimising gate and experts may be insufficiently good to find a clean decomposition.

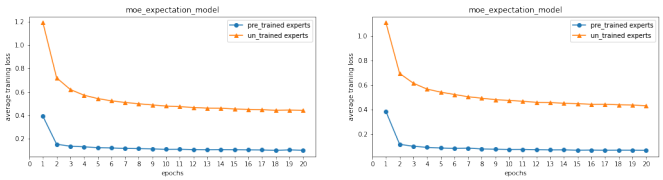
Firstly, let us define interpretable task decompositions for the MNIST dataset and determine if the gate can learn these decompositions. We split the 10 digits into 5 sets of 5 pairs of digits: (i) *{{0,1}, {2,5}, {3,4}, {6,9}, {7,8}}*; (ii) *{{0,3}, {1,2},{4,7}, {5,9}, {6,8}}*; (iii) *{{0,4}, {1,8}, {2,9}, {3,6}, {5,7}}*; (iv) *{{0,5}, {1,9}, {2,6}, {3,7}, {4,8}}*; and (v) *{{0,9},{1,4},{2,7},{3,6},{5,8}}*, avoiding pairs of digits that are usually mis-classified as each other, such as, *{3,8}* and *{4,9}*. Each of the 5 experts was trained with only data samples of one of the 5 pairs of digits. We then fixed the parameters of these pre-trained experts and trained the gate with the same. The gate expert selection tables for splits (i) and (ii), seen in Figure 6 (gate expert selection tables for splits (iii)-(v) are in Appendix A.2)), show that the gate can indeed learn to select the correct expert for each digit and hence learn an interpretable task decomposition.



(i) {{0,1},{2,5},{3,4},{6,9},{7,8}} (ii) {{0,3},{1,2},{4,7},{5,9},{6,8}}

Figure 6: Experts selected by the gate for MNIST test data classification, for each digit, by models trained with pre-trained experts trained on 2 different splits of the 10 digits.

For the sake of completeness, we also wanted to check if: (1) a pre-trained gate trained with un-trained experts; and (2) a pre-trained gate trained with pre-trained experts can train a new set of experts by decomposing the task similar to the experts the gates were learned from. This enables us to check the performance of the gate task decomposition for an interpretable partition vs an un-interpretable partition. The parameters of the pre-trained gate were fixed. We tested cases (1) and (2) with the new expert parameters initialized to: (a) default parameters; and (b) to the same initial parameters of the experts the gates were trained from. In both cases (a) and (b), the gate decomposed the tasks exactly as in Figures 5 and 6 respectively for cases (1) and (2).



(a) (b)

Digits	Pre-trained Experts	Un-trained Experts
(3,8)	**0.02**	0.25
(8,3)	0.50	**0.30**
(4,9)	0.17	**0.05**
(9,4)	0.52	**0.12**
(5,8)	0.18	**0.16**
(8,5)	**0.26**	0.40

(c)

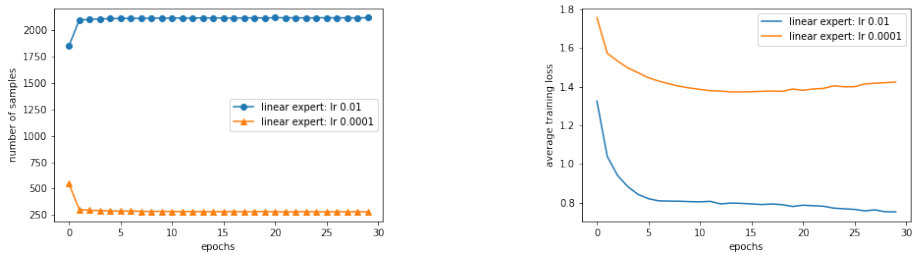
Figure 7: Training loss comparison, for MNIST dataset, for pre-trained gate, trained on pre-trained and un-trained experts and training: (a) experts with default parameter initialization; and (b) experts with the same initial parameter initialization as the experts trained from. (c) Mean negative log likelihood error of mis-classifying commonly mis-classified pairs of digits using pre-trained experts (interpretable model) and un-trained experts (un-interpretable model) for the MoE expectation model.

Let us now check the training loss of the models with pre-trained gates learnt from pre-trained and un-trained experts, training new experts. Figure 7a shows the training loss for cases (1a) and (2a)

and Figure 7b shows the training loss for cases (1b) and (2b), both averaged over 10 runs of the experiment. We see that the interpretable model trained with pre-trained experts converges faster than the un-interpretable model trained with un-trained experts.

We now check the errors made by the two models. Table 7c shows the mean negative log likelihood of mis-classifying a given digit with another digit it is most commonly mis-classified as. For example the negative log likelihood of mis-classifying 3 as 8 for each input sample $x = 3$ is $-\log(p(3|x = 3, \theta)/(p(3|x = 3, \theta) + p(8|x = 3, \theta)))$, where θ is the model parameters. The lower the value the better. Table 7c shows that neither model has a consistently low negative log likelihood error for commonly mis-classified digit pairs and hence, there is no error disadvantage for an interpretable task decomposition.

Lastly, we check if the heuristic of jointly training the gate and expert is sufficiently good for a good decomposition. We tested this by training a MoE expectation model with 2 linear experts with different learning rates, on the toy classification dataset and measured the training loss and sample distribution between them over the epochs (averaged over 10 runs of the experiment), shown in Figure 8. Figure 8b shows the training loss of the 2 experts. We see that the expert with higher learning rate (faster expert) converges faster than the expert with lower learning rate (slower expert). This results in the faster expert grabbing more samples through out the training, starving the slower expert as seen in Figure 8a. Hence, the un-interpretable task decomposition is most likely due to the heuristic of jointly training the expert and gate 'end-to-end' resulting in unequitable data samples allocation between the experts during training.



(a) (b)

Figure 8: Training (a) sample distributions, and (b) loss of 2 linear experts with different learning rates for the toy classification dataset.

In these preliminary experiments we observe that, though clean decompositions do exist with a clear learning advantage and no high error disadvantage, the gate does not learn these clean decompositions when both experts and the gate and jointly trained 'end-to-end'. This could be because some experts learn early, grabbing the training data, starving the other experts of training data.

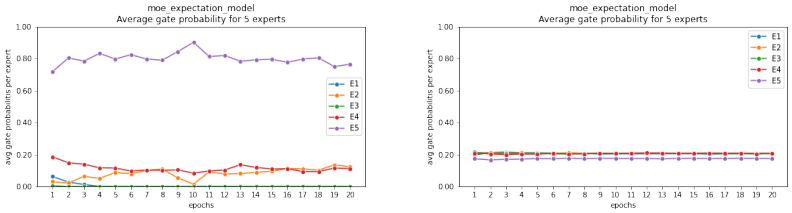
5.1 What are the pathological cases of task decomposition?

The experiments have revealed two pathological cases:

1. **One expert learns all the classes -** This is a known problem and Kirsch et. al. [9] refer to it as *module collapse*. This occurs when the gate selects the same expert for all the samples. In this case the MoE output does not depend on the gate. This could lead to poor interpretability as in effect this is the same as the single model.
2. **One expert learns only one class -** This case has not been previously explicitly reported and occurs when an expert classifies all inputs as the same class. In this case the gate does all the classification and could result in poor transferability.

Both these cases suggest poor allocation of data samples to the experts, by the gate, subsequently leading to either over use or starvation of experts. For interpretability and transferability we need to avoid both these cases. Shazeer et. al. [18] proposed adding a regularization term to the loss that measures the coefficient of variation of the gate output probabilities to avoid module collapse.

We measured the average gate outputs (which are the expert selection probabilities) over all the samples for each epoch with and without this regularization. Gate probability for each expert for each sample is the probability of that expert being selected for that sample. The average gate probability gives an idea of how many times an expert is selected by the gate. Figure 9 shows the average gate probabilities with and without regularization and we see from Figure 9b that the regularization does lead to more equitable utilization of all experts.



(a) Without regularization (b) With regularization

Figure 9: Average gate probabilities per expert for MNIST dataset with and without gate coefficient of variation regularization for equitable expert utilization.

However, an equitable utilization of experts does not necessarily lead to an interpretable task decomposition. Figure 10 compares the expert selection by digits for the model with regularization, with a model trained with one possible interpretable task decomposition. We see in Figure 10a that with regularization the digits are equitably distributed between the experts, though the decomposition is still not as clean as that in Figure 10b of the interpretable task decomposition. For example, for digit 7 the gate uses Experts 1 and 2 and for digit 2 it uses Experts 2 and 3. But we do see that by changing the training dynamics, adding regularization in this case, we can improve the task decomposition.

(a) (b)

Figure 10: (a) Experts used by the gate for classification of each digit with regularization (b) Experts used by the gate for classification of each digit with pre-trained experts for the split *{{0,1}, {2,5}, {3,4}, {6,9}, {7,8}}* for MNIST test data.

6 Discussions and Conclusions

In this paper we have presented various experiments we conducted to better understand how the gate decomposes the tasks between modules in GMNNs and analyze if GMNNs can lead to interpretable models. Our experiments have revealed that GMNNs are indeed inherently interpretable, however, the existing architectures and methods of training them do not necessarily guarantee an interpretable gate decomposition of the tasks among the modules. We also noticed that there is no learning or error disadvantage for the gate to learn an interpretable task decomposition. Hence, it is most likely the heuristic of jointly optimising the gate and experts that leads to uninterpretable task decompositions.

If we could achieve an interpretable task decomposition between the modules then, we can ensure error attribution to either the gate or the modules and additionally the modules would be better suited for transferability.

7 Future Work

We have identified two main problems that could lead to un-interpretable GMNNs: (1) Poor allocation of training examples leading to poor expert utilization and starvation; and (2) unpredictable task decomposition. We are currently working on the following approaches to improve both these conditions:

1. It seems only natural that since we want the experts to learn different sub-tasks we should additionally include a loss term that maximises the diversity of what each expert learns.
2. Train the gate with additional criteria than just inputs. One way of achieving this, independent of the application, would be to augment the gate input with the expert outputs.
3. Decoupling training of the experts and gating by having a double headed architecture. That is, training allocation can be made softer than gating decisions.
4. During the initial part of the training let each expert get a fixed fraction of the data randomly sampled. This should bootstrap the experts smoothly.
5. Try other mechanisms of generating expert weighting by the gate instead of softmax, such as escort transformation [13] because it could have better training dynamics.

References

[1] G. Alain and Y. Bengio. Understanding intermediate layers using linear classifier probes. *ArXiv*, abs/1610.01644, 2017.

[2] S. Amershi, M. Chickering, S. Drucker, B. Lee, P. Simard, and J. Suh. Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI 2015)*. ACM - Association for Computing Machinery, April 2015.

[3] D. H. Ballard. Modular learning in neural networks. In *AAAI'87: Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1*, Seattle, Washington, 1987. AAAI Press.

[4] F. Doshi-Velez and B. Kim. Towards A Rigorous Science of Interpretable Machine Learning. (Ml):1–13, 2017.

[5] F. Fuchs, O. Groth, A. Kosiorek, A. Bewley, M. Wulfmeier, A. Vedaldi, and H. Posner. Neural stethoscopes: Unifying analytic, auxiliary and adversarial network probing. *arXiv*, arXiv:1806.05502, 2018.

[6] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixture of local expert. *Neural Computation*, 3:78–88, 02 1991.

[7] R. A. Jacobs, M. I. Jordan, and A. G. Barto. Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive Science"*, 15(2):219 – 250, 1991.

[8] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Proceedings of the International Joint Conference on Neural Networks*, 2:1339–1344, 1993.

[9] L. Kirsch, J. Kunze, and D. Barber. Modular networks: Learning to decompose neural computation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[10] C. A. Knoblock. Learning abstraction hierarchies for problem solving. In *Proceedings of the Eighth National Conference on Artificial Intelligence - Volume 2*, AAAI'90, pages 923–928. AAAI Press, 1990.

[11] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010.

[12] Z. C. Lipton. The mythos of model interpretability. *Communications of the ACM*, 61(10):35–43, 2018.

[13] J. Mei, C. Xiao, B. Dai, L. Li, C. Szepesvári, and D. Schuurmans. Escaping the gravitational pull of softmax. *NeurIPS 2020*, pages 1–11, 2020.

[14] D. Mihai and A. Lascarides. *Combining a Mixture of Experts with Transfer Learning in Complex Games*. AAAI Press, Stanford, 2017.

[15] C. O'Neil. *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Crown Publishing Group, USA, 2016.

[16] M. F. Pradier, J. Zazo, S. Parbhoo, R. H. Perlis, M. Zazzi, and F. Doshi-Velez. Preferential mixture-of-experts: Interpretable models that rely on human expertise as much as possible, 2021.

[17] M. Ryabinin and A. Gusev. Towards crowdsourced training of large neural networks using decentralized mixture-of-experts. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3659–3672. Curran Associates, Inc., 2020.

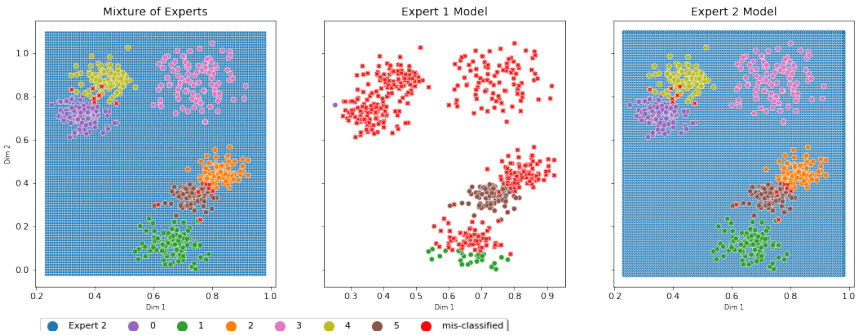
[18] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017.

[19] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1-2):181–211, Aug. 1999.

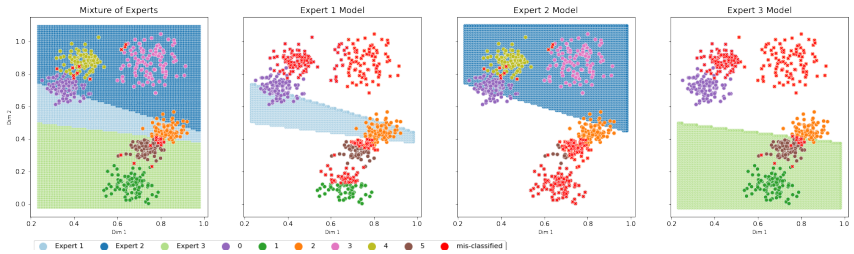
[20] N. Xie, G. Ras, M. van Gerven, and D. Doran. Explainable Deep Learning: A Field Guide for the Uninitiated. 2020.

A Appendix

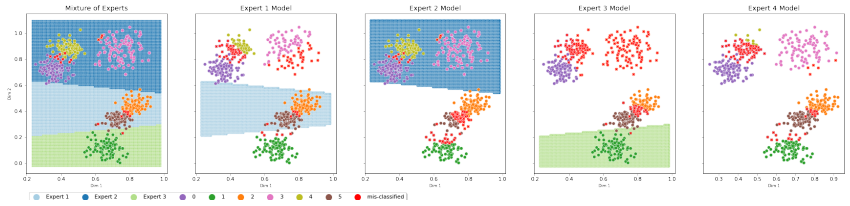
A.1 Toy classification task outputs for EM, pre-softmax and stochastic



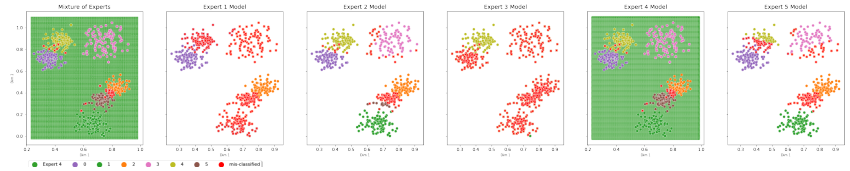
(a) 2 experts



(b) 3 experts

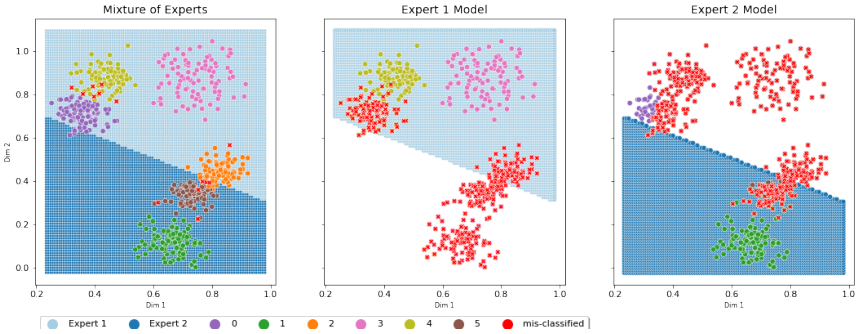


(c) 4 experts

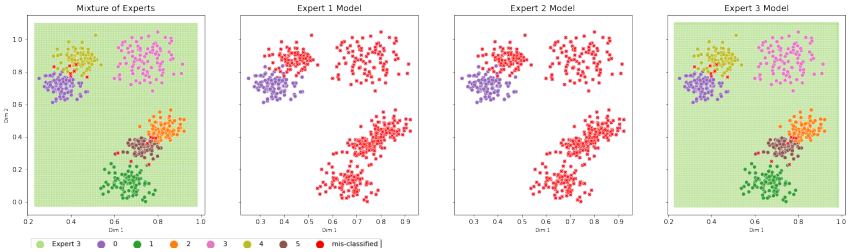


(d) 5 experts

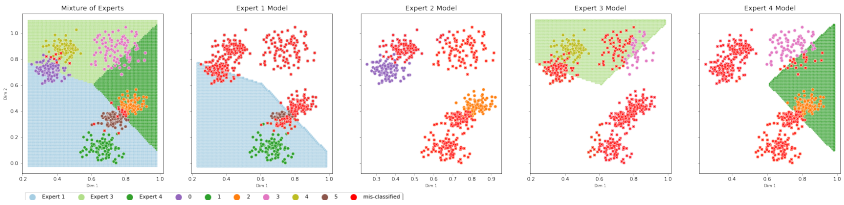
Figure 11: Toy classification test data class predictions by the trained MoE EM model and the individual experts along with the samples gated to the expert by the gate (shaded regions). Red 'x' are the mis-classified samples.



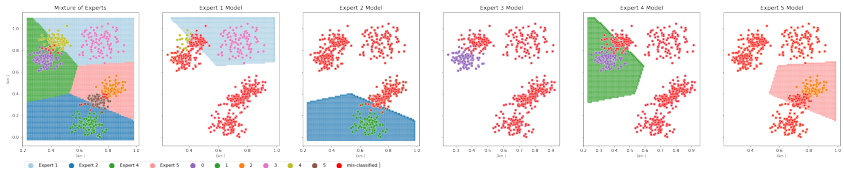
(a) 2 experts



(b) 3 experts

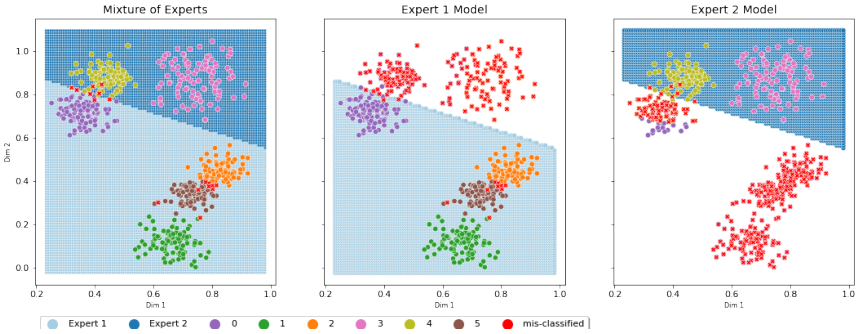


(c) 4 experts

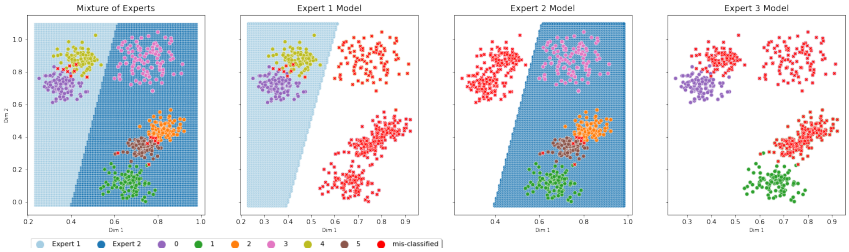


(d) 5 experts

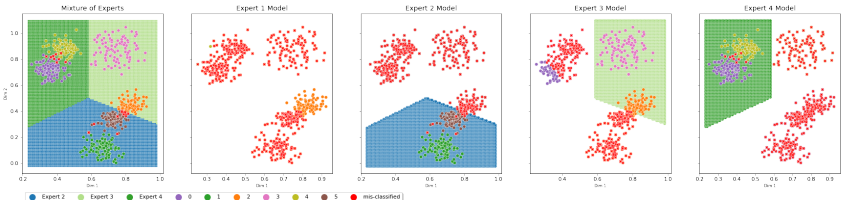
Figure 12: Toy classification test data class predictions by the trained MoE pre-softmax model and the individual experts along with the samples gated to the expert by the gate (shaded regions). Red ‘x’ are the mis-classified samples.



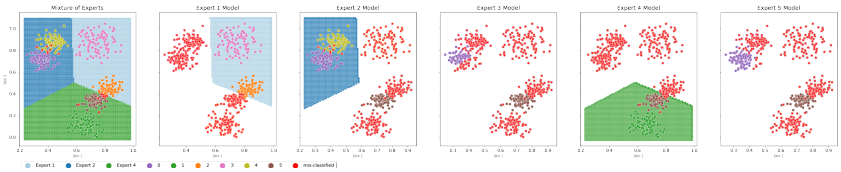
(a) 2 experts



(b) 3 experts

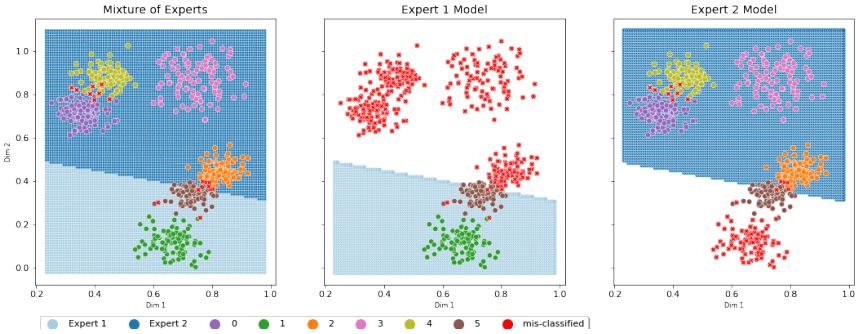


(c) 4 experts

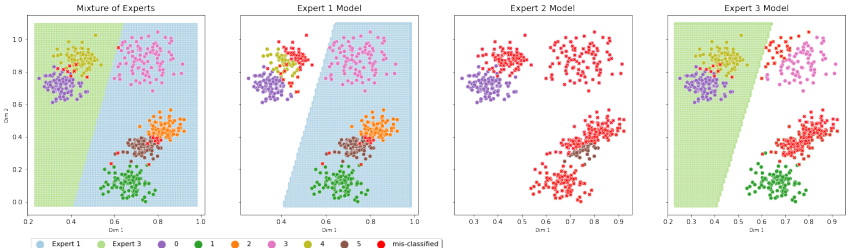


(d) 5 experts

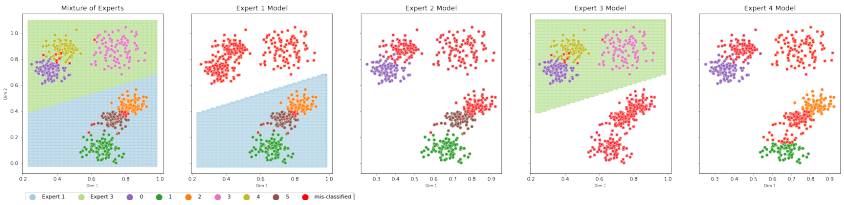
Figure 13: Toy classification test data class predictions by the trained MoE expectation model and the individual experts along with the samples gated to the expert by the gate (shaded regions). Red ‘x’ are the mis-classified samples.



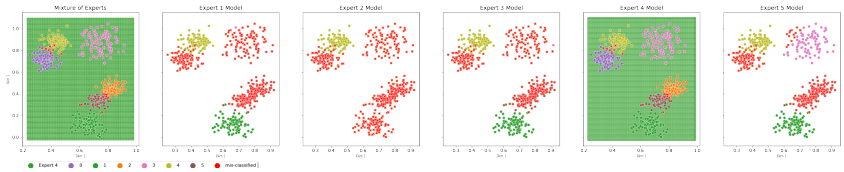
(a) 2 experts



(b) 3 experts



(c) 4 experts



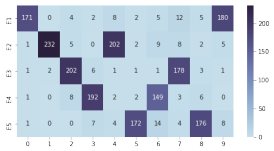
(d) 5 experts

Figure 14: Toy classification test data class predictions by the trained MoE stochastic model and the individual experts along with the samples gated to the expert by the gate (shaded regions). Red 'x' are the mis-classified samples.

A.2 Gate expert selection tables for gate trained with pre-trained experts

(i) {{0,4}, {1,8}, {2,9}, {3,6}, {5,7}}

(ii) {{0,5}, {1,9}, {2,6}, {3,7}, {4,8}}



(iii) {{0,9},{1,4},{2,7},{3,6},{5,8}}

Figure 15: Experts selected by the gate for classification of each digit for 3 different splits of the MNIST dataset.