Multi-Task Deep Representation Learning

Shobhit Chaurasia, Mike Feilbach, Tyler McDonnell, Harshal Priyadarshi
{shobhit, mfeilbach, tmcdonnell, harshal.priyadarshi}@utexas.edu
Department of Computer Science
University of Texas at Austin

ABSTRACT

Multi-task representation learning (MTRL) is a flavor of representation learning — and more specifically, a type of deep learning — in which raw input data is fed to a deep learning network to predict multiple tasks simultaneously. Here, a task is defined as some output or prediction of the input data. Predicting multiple tasks simultaneously may allow for higher prediction accuracy. This is likely due to the ability of a network to leverage similarity and correlation between tasks. For example, when given the same input examples, predicting two related tasks at the same time may provide higher accuracies than predicting the same two tasks independently. We aim to investigate the following questions regarding multi-task learning. First, is multi-task learning beneficial? Second, how does training size impact efficacy of multi-task learning? Third, how does task coupling affect performance? Fourth, how does the degree of sharing impact performance, and how is the degree of sharing linked to task coupling?

1. INTRODUCTION

Multi-task representation learning (MTRL) falls under the category of deep learning, which more broadly falls under the umbrella of representation learning. In a non-representation approach, input to a machine learning algorithm is given in the form of hand-crafted features, which are attributes of input examples. Given input examples, key features which aim to distinguish examples from each other must be identified and evaluated, a traditionally arduous task often requiring domain experts. Representation learning is an approach in which raw data, such as images, text, or signals, rather than features describing them, are fed directly into a machine learning algorithm. This means that the task of identifying hand-crafted features is not necessary — the input is left relatively untouched and does not require the user to determine which features within the input examples will give the machine learning algorithm the best results. In a representation learning system, these features are learned by the algorithm itself, allowing for an "end-to-end" experience where raw input data is fed to the system to produce traditional machine learning outputs. While most representation learning systems rely and thrive on large data sets, it is not always the case that such large data sets exist. For example, in [13], prediction of multiple demographic attributes is explored using raw shopping history as input. The shopping history data was derived from the BeiRen[1] dataset, where

[1]http://www.brjt.cn/

Figure 1: Million Songs Dataset : meta-labels like genre etc. alongside bag-of-words (BOW) lyrics

only 8.96% of users offered their educational background. In this experiment, the BeiRen data set was too sparse to predict any single demographic attribute alone, such as educational background, gender, or marital status; predicting any one of these tasks independently was too difficult given the sparsity of the data. However, when combining the prediction of these tasks together (using MTRL), [13] found it possible to train a successful deep learning network by leveraging correlation between tasks. In [13], tasks such as educational background, which suffered from sparse data labels, were predicted more accurately when being considered alongside other tasks, many of which also suffered from sparsity.

In our research, we plan to utilize a similar approach to train a MTRL network. The following research questions will be considered in this paper:

- **RQ1**: Is multi-task learning beneficial?
- **RQ2**: How does training data size impact the efficacy of multi-task learning? (We expect multi-task learning to provide greater relative benefits when the amount of training data for a target task is small.)
- **RQ3**: How does task coupling impact the performance of multi-task learning? (We expect that shared representations among tightly coupled tasks will offer greater predictive accuracy than those shared between loosely coupled tasks.)
- **RQ4**: How does the degree of sharing impact performance, and how is this linked to task coupling? (We expect that sharing more layers between tasks will be beneficial when the tasks are tightly coupled.)

To investigate these questions, we have built multiple different MTRL networks to test our theories using the Million Song Dataset (MSD) from [4], as detailed in Section 4.

2. RELATED WORK

Since labeled data is hard to obtain in many interesting tasks where machine learning models can be leveraged, there have been numerous attempts to train models with limited data. Common techniques include Domain Adaptation, Transfer Learning, training through Weak Supervision, and Multi-task Learning. Of particular interest in the Deep Learning community is multi-task learning through MTRL.

In the Deep Learning space, MTRL has taken the form of learning a shared representation across potentially related tasks. Three RNN architectures based on multi-task framework were proposed in [9] to model text for a set of related text classification tasks including movie reviews ratings, subjectivity assessment. Joint training of a deep network using the multi-task framework was used in [6] for a host of NLP tasks — POS tagging, NER, chunking, and semantic relatedness of words. In the IR domain, a multi-task deep neural network was proposed in [10] that learns a shared representation for two weakly related tasks — query classification, and document ranking for web-search. The motivation underlying these research efforts is to leverage data from related tasks in order to provide a form of weak supervision to each other for enhanced learning.

It is worth noting that the concept of multi-task learning predates the recent wave of deep neural networks. Multi-task learning was posed as a joint optimization problem in [3], and was used alongside CRFs in [2] to improve chunking in semantic parsing. In the domain of Music IR — which is used in this work (see Section 4) — a semantic embedding space is induced in [14] (through a non-neural network based method) by jointly optimizing on multiple music-related prediction tasks such as artist, song, and tag prediction; and retrieval tasks such as retrieving similar songs and artists.

3. METHODOLOGY

Conventional methodology for solving multiple prediction tasks involves constructing separate models for each task. This approach, while straightforward, fails to leverage potential correlations across these tasks. Different prediction tasks — especially the ones that are closely related — can aid each other by making use of a shared representation of the input. To this end, we investigate three deep neural network architectures for multi-task learning alongside a single-task variant, all of equal depth and equal layer widths.

3.1 Network Architectures

3.1.1 Low-level sharing

The proposed architecture is shown in Fig. 2(b). The idea is borrowed from [13]. Each task influences the other by updating the common weight parameters in the shared layers. The weight sharing happens at the lower levels, which goes with the intuition that related tasks are coupled together through a common low-level representation — a form of a shared knowledge-base — and each task builds upon the shared low-level representation in its own unique way.

3.1.2 High-level sharing

The proposed architecture is shown in Fig. 2(c). The two tasks learn independent low-level representations which are combined together in higher-level layers through shared weights. As in the network in Fig. 2(b), each task builds upon the shared representation in its own unique way.

3.1.3 Interspersed sharing

The proposed architecture is shown in Fig. 2(d). This is a combination of the previous two architectures: the architecture learns low-level as well as high-level shared representations, separated by independent mid-level representations. The idea of blending the previous two architectures can be extended to construct architectures with shared weights interspersed throughout the network. However, we choose to limit our study of interspersed sharing to the architecture shown in Fig. 2(d) due to time limitation, and in the interest of interpretability of results — something that is hard in an architecture with truly interspersed sharing.

3.1.4 No Sharing

The single-task architecture used as the baseline in our study is shown in Fig. 2(a).

3.2 Key Architecture Decisions

The following architectural decisions are to be made that are common for any deep neural network.

3.2.1 Network Architecture

There are essentially three types of deep network architectures that can be relevant here:

1. Deep Neural Network
2. Deep Stacking Network
3. Deep Convolution Neural Network

However, CNN's are not relevant for our dataset as we have a bag of words lyrics representation, and thus no spatial context is available in 1-D dimension. It is not very clear how we can employ sharing of layers for DSN, as we would have to incorporate sharing on each stack, and usually each stack of a Deep Stacking Network is not very deep and usually has just one hidden layer. The sharing of stack is not principled w.r.t. our approach as we want sharing across representations, while there is no incentive that the higher level stack learns higher representation than the lower level stack in a DSN. Thus we plan to proceed with the Deep Neural Network Architecture itself.

3.2.2 Network Depth and Width

The depth of the network is 5, with 4 fully connected hidden layers and an output layer. The choice of depth was chosen using validation testing and time profiling, in order to prevent overfitting and simultaneously address the time limitation of the project. The width of the hidden layers were fixed to be $(fc1 : 1024, fc2 : 512, fc3 : 256, fc4 : 128)$. This decision was mutually agreed upon, and is not backed by experimental validation.

3.2.3 Activation Function

The activation function we chose is a RELU network as it prevents to a certain degree the problem of saturation during back-propagation, and is a common choice for this network.

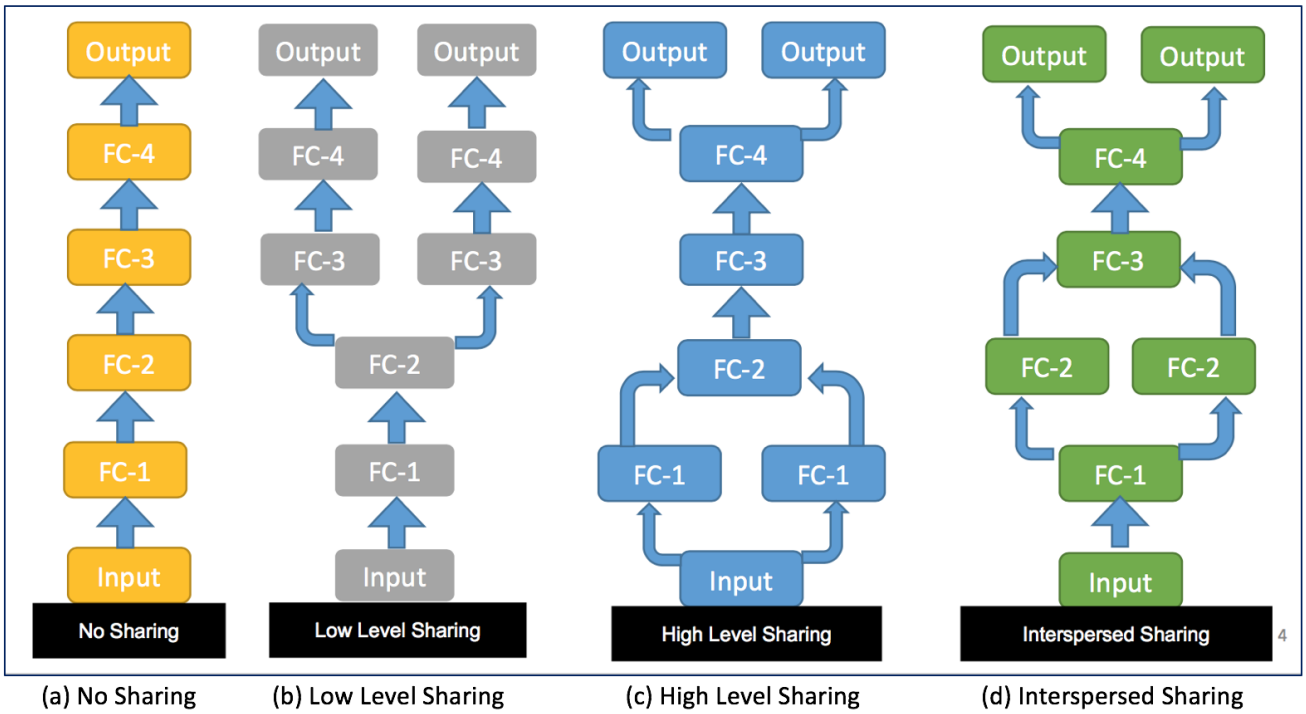

Figure 2: Deep Neural Architecture with individual and sharing variants

3.2.4 *Input Preprocessing*

We will be preprocessing the input values to be mean normalized so that the final input has zero mean and unit variance. This is called **whitening**, and has been shown to improve learning and also be more robust to initial weight initialization.

3.2.5 *Objective Function*

The tasks that we have in hand are a mix of regression and classification. For instance labels corresponding to sub-tasks like prediction of Year of Composition and Key of the song, have discrete labels, while others like "Hotttnesss", "Danceability" etc. take continuous values.
The objective function chosen for Classification task is **Softmax Cross Entropy Loss**, while the objective of the Regression task is **Mean Square Error Loss**

3.2.6 *Optimization Algorithm*

The optimization algorithm will be Mini-batch Stochastic Gradient Descent[5] algorithm. This is the default choice that has been widely reported by the Deep Learning community to be the best performing, and swift to train. We add mini-batching to ensure that the gradient calculation at each stage is as close as possible to the true gradient. Mini-batching also improves parallelization and thus reduces training time.

3.2.7 *Optimization Update Type*

The optimization update can be done by various ways once we have the gradient corresponding to each parameter to update. The usual choices of update routines are:

1. Ada-Grad
2. Ada-delta[15]
3. Adam[8]
4. Momentum[12]

We used Ada-Grad optimizer, as they are reported by Deep learning community to quickly escape saddle points and ensure faster convergence, and thus are better for Deep Neural Network.

3.2.8 *Regularization Type*

These are the regularization types that is widely reported as useful in training deep learning algorithms and preventing overfitting.

1. Dropout[11]
2. Batch Normalization[7]
3. L2 Normalization

It has been shown in [7], that Batch normalization improves learning for higher learning rate, prevents divergence, and at the same time provides regularization. It has also been shown that Batch normalization can effectively replace Dropout. However, it adds extra cycles of compute to the forward propagation. Thus we adopted **Dropout** as the regularization objective.

3.2.9 *Implementation Details*

We decided to choose Tensorflow [1] to implement deep learning, for ease of coding the model and intuitive interface with extensive documentation.The training and evaluation were performed on CPU as well as Maverick GPU supercomputing platform.

All our models and experiments are made available online.[2]

4. EVALUATION

4.1 Experimental Setup

For evaluation, we adopt the MILLION SONG DATASET (MSD) [4]. This dataset consists of features derived either directly from the audio signal of a song (e.g., duration) or classification labels provided by human annotators (e.g., genre tags). It is interesting to note that these features are in some sense derived explicitly from the *audio signal* of a song, not the lyrics. Nevertheless, we treat these as *textual* prediction tasks. In this sense, we assume that there is a relationship between the lyrics of a song and characteristics of its final musical form, such as its genre, the year it was released, or loudness of the song. For some simple intuition on why this is a valid assumption, consider that songs which explore universal themes such as love and loss tend to be relatable to a larger body of listeners, and thus more popular than those which explore niche ideas. Moreover, in this study we are concerned with exploring the effects of learning across multiple tasks, rather than providing state-of-the-art performance for a particular prediction task. To this end, we feel these assumptions are acceptable.

Since our research interest is in *multi-task* learning, we require multiple prediction tasks which all arise from the same input data in order to evaluate our research questions. To this end, we have selected a subset of the features provided for each song in the MSD to serve as independent *prediction tasks*. For each feature selected, the prediction task will be to predict the provided feature label given the lyrics of a song as input in a bag-of-words (BOW) format. The BOW for each song was obtained by parsing the musiXmatch dataset, a sister set to MSD [4]. A full description of all prediction tasks used throughout the evaluation can be found in Table 1. Reasoning as to why these tasks were chosen is included in the sections that follow.

Though the MSD consists of data for one million songs, not all songs are labeled with all relevant features. There are approximately 60,000 songs for which all features are populated, so we use these songs as the basis for our experiments. We sub-divided this data into three sets: (1) a training set consisting of 25K songs; (2) a validation set consisting of 6K songs; and (3) a test set consisting of 25K songs.

Throughout each of the experiments that follow, we compare the performance of the four architectures described in Section 3 under a variety of circumstances. In each case, we evaluate multi-task architectures against one another and against the traditional single-task architecture by comparing performance on a *target task* common to each model being evaluated. Each instance of a single task model will be trained exclusively on examples from the target task, whereas multi-task models will be trained on the target task and adjacent tasks, which may be either closely related (tightly coupled) to the target task, or unrelated.

For some experiments, we vary the number of training examples, always pulling from the training set of 60K songs. All models are evaluated on the same validation set. Results are reported for the validation sets, either in the form of a learning curve or final accuracy. We originally intended to evaluate each of the models on the hold-out test set after training, but the training took much longer than our pilot experiments suggested. It is worth noting that our validation set can indeed be viewed as a held-out set in disguise since, owing to time constraints, we could not use it for hyperparameter tuning. Unlike our original plans, the performance on the validation set did not end up influencing our architectures and training procedure, and hence, numbers reported on the validation set can be treated as a proxy for models' performance on unseen data.

To reiterate the central goal of multi-task learning, we expect that training multi-task networks using examples for adjacent, non-target tasks will result in stronger predictive accuracy for the task of interest. Our evaluation explores the following research questions in turn:

- RQ1: Is multi-task learning useful?
- RQ2: How do task coupling and network architecture affect performance?
- RQ3: How does training size impact performance?

4.2 RQ1: Multi-Task Learning

Our first research question is to validate the frequent, but unsupported, claim that representations can be *usefully* shared across tasks in deep neural networks. To this end, we simply selected one of the model architectures, the low-level sharing architecture, from 3 and apply it using "arbitrary" (not really arbitrary, but we will discuss this more soon!) tasks: {POP, LOUDNESS, YEAR}. In this case, the target is the POP task, so we also train and evaluate a single-task model for comparison. We trained both the multi-task and single-task model on 1,000 examples. We select a very small number of examples by deep learning standards, because we believe this is where multi-task learning is likely to be the most useful in practice: when there is limited availability of training data for the desired task.

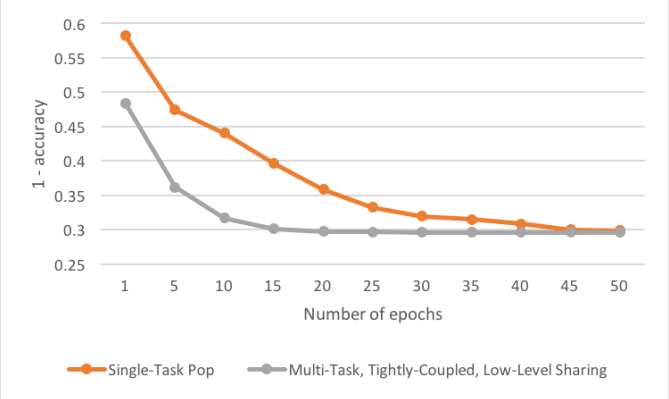

Figure 3: Baseline Performance

The results of this experiment are shown in Figure 3. Each data point represents the results of evaluating the network over the validation data set after the specified number of training epochs. As we can see, both the single-task and multi-task models quickly converge to the same performance, and neither outperforms the other from that point. Notably, the multi-task model does converge in fewer epochs.

[2]https://github.com/harpribot/representation-music

Number	Feature	Type	Description
1	Loudness	Regression	Real number representing the general loudness of an audio track.
2	Year	Regression	Real number representing the year in which a track was released.
3	Pop	Classification	Boolean representing whether a song is pop.
4	Pop Rock	Classification	Boolean representing whether a song is pop rock.
5	Ballad	Classification	Boolean representing whether a song is a ballad.

Table 1: Task Descriptions

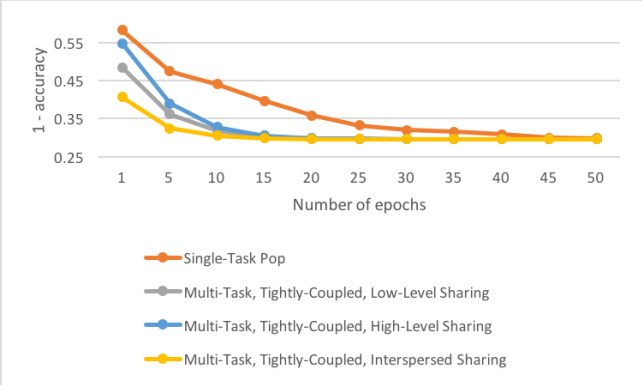

Figure 4: Tightly Coupled Performance

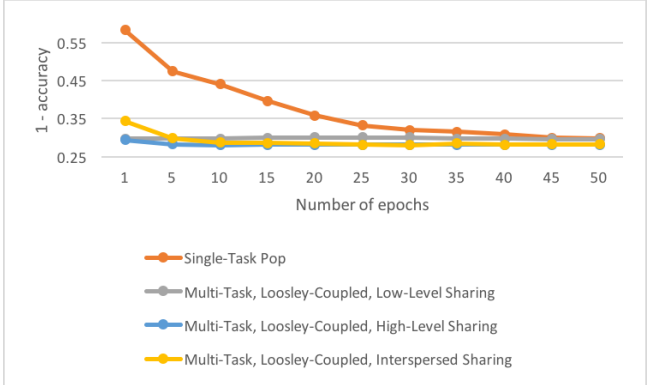

Figure 5: Loosely Coupled Performance

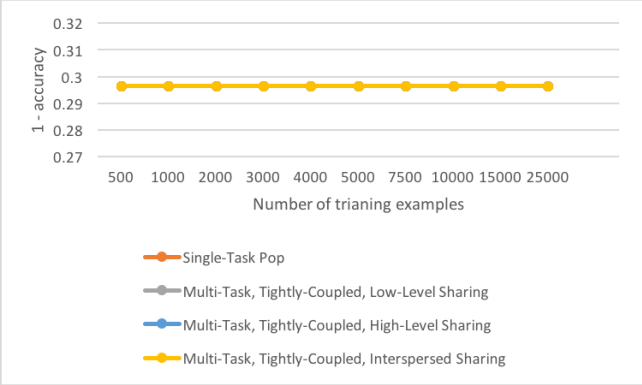

Figure 6: Tightly Coupled - Training

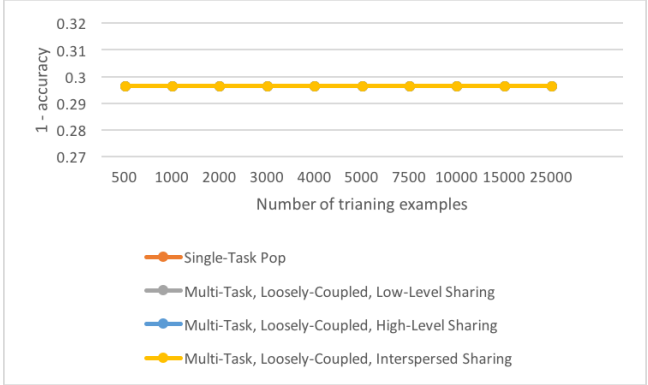

Figure 7: Loosely Coupled - Training

These results may be discouraging, but one positive note is that multi-task learning does not appear to *harm* accuracy. Additionally, since we have arbitrarily selected tasks, perhaps the given architecture does not align with the relatedness of the tasks, or perhaps the tasks are simply not related at all. Indeed, these questions are important to our next research question.

4.3 Task Coupling & Architecture

Our next set of experiments focus on the question of how task coupling (i.e., the relatedness of tasks) ties into the successes of multi-task learning, and in particular, the selection of deep architecture. For these experiments, we operate on two sets of tasks: a *tightly coupled* set, consisting of tasks which are highly related; and a *loosely coupled* set, consisting of tasks which are only marginally related. In order for us to construct such sets of tasks, we need to take a step back to discuss and formalize the concept of task relatedness.

4.3.1 Task Relatedness

Deep neural networks build representations over raw input data over many layers, where deeper layers in the network tend to correspond to higher-level representations of the data. Multi-task learning operates under the observation, or assumption, that even for disparate tasks with identical input formats, such networks may learn representations common to different features. For example, one could imagine that the lower levels of two deep neural networks for identifying cats and dogs in images might share common low-level representations corresponding to edges, with only deeper layers concerned with higher-level abstractions unique to a cat or dog, respectively. In particular, the assumption being made is that different tasks can share similar structures in some abstract sense. Unfortunately, though this notion may be intuitive, it is difficult to quantify, just as it is difficult to interpret the hidden layers of deep networks.

Instead, we develop an approximation to this abstract sense of similar representations using task correlation. In particular, we assume that tasks with high Pearson correlation coefficients (i.e., highly correlated) are likely to share similar representations in the abstract sense. We use this assumption to build two datasets which we use throughout the rest of evaluation, a *tightly coupled* and *loosely coupled* set. To build these, we selected a task at random, POP, from the 2000 tasks available in our dataset, and computed the pairwise Pearson correlation between POP and every other task in the dataset. To form our set of tightly coupled tasks, we select the two most highly correlated tasks from these computations: POP ROCK and BALLAD. To develop our loosely coupled set, which we simply select two tasks at random for which there is little positive or negative correlation, LOUDNESS and YEAR. To summarize, the sets of tasks we use are as follows:

- **Tightly Coupled**: {POP, POP ROCK, BALLAD}
- **Loosely Coupled**: {POP, LOUDNESS, YEAR}

4.3.2 Experiment

To evaluate how task coupling may affect multi-task learning and the selection of deep architecture, we evaluate the high-level sharing, low-level sharing, and interspersed architectures for both the tightly coupled and loosely coupled set of tasks and compare performance on the POP task. Our hypothesis is that multi-task learning will perform better for tightly coupled tasks, since the learned representations are likely to be more similar. Additionally, we hypothesize that high-level sharing architectures will be more useful for tightly coupled tasks, and low-level sharing architectures for loosely coupled tasks.

Figure 4 and Figure 5 show the results for the tightly coupled and loosely coupled tasks trained across all four models using 1,000 training examples. Surprisingly, all models, regardless of the degree of the task coupling or model architecture used, appear to converge to the same accuracy. This is especially surprising, since 1,000 examples is very few for training a deep network, when we would expect multi-task learning to be particularly effective. Again, we do see that multi-task models converge with fewer epochs than the single-task model, but ultimately provide no asymptotic performance gains.

4.4 RQ3: Training Size

Our final set of experiments looks at how the amount of training data affects the performance of multi-task learning. Though this is interesting as an independent research question, following the disappointingly uniform results in the previous section, it is also a way of "debugging" why MTRL has not been as effective as we hypothesized. For these experiments, we follow the same setup in Section 4.3, but evaluate each model trained upon different number of training examples to compare how multi-task learning and single-task learning evolve with greater availability of data.

Figure 6 and Figure 7 show the results for different numbers of training examples. Once again, even across the range of training data sizes tested, we do not see any interesting developments.

5. DISCUSSION

In this paper, we explored deep multi-task representation learning, in which a deep neural model learns to provide predictions for several tasks at once. A common claim throughout the deep learning literature is that abstract representations learned by deep neural networks can be shared across a variety of tasks. Despite this, there is relatively little literature which attempts to quantify the benefits of doing so. In this paper, we have done exactly this, and explored how variables such as amount of training data, task coupling, and network architecture play into successful multi-task learning.

Unfortunately, our results were not very encouraging. We saw virtually no improvements in asymptotic performance from multi-task approaches using any of the network architectures presented in the literature, or different task couplings. In fact, we did not even see these results vary significantly as we increased training data sizes. These results are disappointing, as multi-task learning fits strongly with our intuition: humans are very successful at sharing knowledge abstractions usefully across related domains. Since deep learning is capable of learning abstract feature representations reminiscent of human processing, and is also more data hungry than alternative flavors of machine learning, we expect it to be a strong candidate for multi-task learning.

Despite this, we are hesitant to draw any strong negative conclusions about multi-task representation learning from the results presented here for a number of reasons. First and foremost, we never actually had the opportunity to evaluate the final learned models on the test sets we had set aside. Instead, all results presented were produced on the validation set. In our case, the validation set was still a hold-out set, since we did not perform hyperparameter tuning (which could be another reason for poor performance), but it is still undesirably small, containing only 6,000 examples. This could explain why each and every model converged to the same accuracy: perhaps the validation set was not an adequate sampling of the data, and since it was so sparse, it was difficult to improve on the "tough" examples with such limited training data. We might have seen much more variation with our desired test sets. This also brings up the notion of training data. Deep learning architectures are notoriously data hungry, and it is possible that in our quest to explore the utility of multi-task learning under severe training data constraints, we used inadequate amounts of training data.

Additionally, there are potential issues with the dataset used. One issue which we explored was that the scale of outputs that were being predicted for distinct tasks (e.g., LOUDNESS vs. YEAR vs. the boolean POP) were vastly different. Though it is common to pre-process the input to machine learning algorithms, it is not common to do so with the predictions. However, this is a special case where the scale of one task may affect other tasks through backpropagation. It is unclear if this could affect results, so we did try repeating these experiments after normalizing the *outputs* of the prediction tasks, but we did not see any notable gains. A second issue might be related to the bag-of-words input. Ideally, we would have liked to use the actual sequence of lyrics as input, as we expect this to carry much more information usable by deep neural networks, but unfortunately, we were unable to obtain this data. It is possible the BOW lacked sufficient discriminative information for these tasks.

Finally, it is also unclear that our assumptions about task relatedness (see Section 4.3.1) are sound. In our formulation of tightly and loosely coupled tasks, we assume that

tasks which are highly correlated are amenable to shared representations. Indeed, tasks which are not correlated at all may still share similar useful abstractions, while highly correlated tasks may not. As an example, consider again two deep neural networks trained to recognize distinct and unrelated objects in images. Regardless of what the objects are, the lower layers of a convolutional neural network trained for these tasks are likely to learn representations related to common concepts such as edges. Thus, perhaps our formulation of correlation is misleading and the task sets which we have defined are in fact not related in the ways we assume.

6. FUTURE WORK

With these numerous concerns enumerated, there are many opportunities for future improvements to this work. For one, evaluation of the learned models should be done on a separate, large, and representative test set, or perhaps on a different dataset altogether. These experiments would also benefit from an examination of representation sharing that characterizes task relatedness in a more principled manner.

Separately, future work might conduct the training size experiments in an alternate manner. Rather than varying the number of training examples used from all tasks simultaneously, it would be useful to understand if multi-task learning sees performance increases on a target task by training on a larger number of training examples from *non-target tasks*. In fact, this is scenario is more likely to occur in practice, where we may have many examples from one domain which could be used to transfer knowledge to an adjacent domain.

Finally, another avenue worth exploring is an alternate strategy for training multi-task networks: instead of backpropagating the combined error from all the tasks simultaneously, the network can be trained by alternating between errors on individual tasks for backpropagation. In the latter strategy, each task individually gets a chance to directly influence the others.

7. ACKNOWLEDGMENT

The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC, visualization, database, or grid resources that have contributed to the research results reported within this paper. URL: http://www.tacc.utexas.edu

8. REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org*, 1, 2015.

[2] A. Agarwal and S. Kataria. Multi-task crf model for predicting issue resolution status in social media based customer care.

[3] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853, 2005.

[4] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[5] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[6] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[7] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[8] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] P. Liu, X. Qiu, and X. Huang. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*, 2016.

[10] X. Liu, J. Gao, X. He, L. Deng, K. Duh, and Y.-Y. Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proc. NAACL*, 2015.

[11] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[12] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton. On the importance of initialization and momentum in deep learning. *ICML (3)*, 28:1139–1147, 2013.

[13] P. Wang, J. Guo, Y. Lan, J. Xu, and X. Cheng. Multi-task representation learning for demographic prediction. In *European Conference on Information Retrieval*, pages 88–99. Springer, 2016.

[14] J. Weston, S. Bengio, and P. Hamel. Multi-tasking with joint semantic spaces for large-scale music annotation and retrieval. *Journal of New Music Research*, 40(4):337–348, 2011.

[15] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

APPENDIX

A. DIVISION OF WRITING

The writing part of the paper is divided as follows:

- **Abstract + Introduction**: Mike Feilbach
- **Related Work + Network Architectures**: Shobhit Chaurasia
- **Network Architecture + Key Architecture Decisions**: Harshal Priyadarshi
- **Evaluation**: Tyler McDonnell
- **Discussion**: Harshal Priyadarshi + Tyler McDonnell
- **Future Work**: Tyler McDonnell + Shobhit Chaurasia

B. DIVISION OF WORK

- Shobhit Chaurasia
  - Initial literature review to propose candidate network architectures and research questions worth exploring in our study.
  - Implementation of training, validation, checkpointing, and held-out evaluation (which we ended up not doing).
  - Implementation of an interface to the skeleton TensorFlow code built by Harshal to facilitate experimentation.
  - All other tasks done jointly with Harshal (see Harshal's section for details).
- Mike Feilbach
  - Researched datasets, presented Million Song Dataset.

  - Researched and partitioned subset of Million Song Dataset for experiment based on constraints.
  - Wrote back-end code to partition dataset and generate BOW lyrics (given to Tyler to build upon).
  - Worked with Tyler to update back-end code based on requirements (e.g., updated back-end code to support binary labels for genres).
  - Assisted Tyler to plan final experiments.
  - Parsed and graphed final experiment results.
- Tyler McDonnell
  - Initial definition of dataset requirements and early investigation of datasets.
  - Preprocessing of 300GB of dataset files to extract and clean-up features for prediction tasks. Transformed categorical genre tags into binary classification tasks and interface for accessing data.
  - Devised and ran pilot experiments. Performed correlation analysis to devise final experiments.
  - Devised and wrote final experiments.
  - Interpretation of results.
- Harshal Priyadarshi
  - Coded the entire Deep Neural Network Model.
  - Performed Experiments for the individual models as well as the shared representation models. Shobhit and Harshal have performed majority of the experiments.
  - Fixed the architecture hyper-parameters together with Shobhit Chaurasia.
  - Fixed the four different shared architectures with Shobhit Chaurasia and coded them with Shobhit.
  - Fixed the experiments to perform with Tyler Mcdonell, and Shobhit Chaurasia
  - Have maintained and re-factored the entire project including experiments and models for scalability and readability.
  - Shobhit and my contribution towards the codebase for the experiment is very high, and we would like this to be considered for grading purposes. The contributions can be tracked based on the commit logs on the github repository.