When to Use Multi-Task Learning vs Intermediate Fine-Tuning for Pre-Trained Encoder Transfer Learning

Orion Weller* Johns Hopkins University
Kevin Seppi Brigham Young University
Matt Gardner Microsoft Semantic Machines

Abstract

Transfer learning (TL) in natural language processing (NLP) has seen a surge of interest in recent years, as pre-trained models have shown an impressive ability to transfer to novel tasks. Three main strategies have emerged for making use of multiple supervised datasets during fine-tuning: training on an intermediate task before training on the target task (STILTs), using multi-task learning (MTL) to train jointly on a supplementary task and the target task (pairwise MTL), or simply using MTL to train jointly on all available datasets (MTL_{All}). In this work, we compare all three TL methods in a comprehensive analysis on the GLUE dataset suite. We find that there is a simple heuristic for when to use one of these techniques over the other: pairwise MTL is better than STILTs when the target task has fewer instances than the supporting task and vice versa. We show that this holds true in more than 92% of applicable cases on the GLUE dataset and validate this hypothesis with experiments varying dataset size. The simplicity and effectiveness of this heuristic is surprising and warrants additional exploration by the TL community. Furthermore, we find that MTL_{All} is worse than the pairwise methods in almost every case. We hope this study will aid others as they choose between TL methods for NLP tasks. [1]

1 Introduction

The standard supervised training paradigm in NLP research is to fine-tune a pre-trained language model on some target task (Peters et al., 2018; Devlin et al., 2018; Raffel et al., 2019; Gururangan et al., 2020). When additional non-target supervised datasets are available during fine-tuning, it is not always clear how to best make use of the supporting data (Phang et al., 2018, 2020; Liu et al., 2019b,a; Pruksachatkun et al., 2020a). Although there are an exponential number of ways to combine or alternate between the target and supporting tasks, three predominant methods have emerged: (1) fine-tuning on a supporting task and then the target task consecutively, often called STILTs (Phang et al., 2018); (2) fine-tuning on a supporting task and the target task simultaneously (here called pairwise multi-task learning, or simply MTL); and (3) fine-tuning on all N available supporting tasks and the target tasks together (MTL_{All}, $N > 1$).

Application papers that use these methods generally focus on only one method (Søgaard and Bingel, 2017; Keskar et al., 2019; Glavas and Vulić, 2020; Sileo et al., 2019; Zhu et al., 2019; Weller et al., 2020; Xu et al., 2019; Chang and Lu, 2021), while a limited amount of papers consider running two. Those that do examine them do so with a limited number of configurations: Phang et al. (2018) examines STILTS and one instance of MTL, Changpinyo et al. (2018); Peng et al. (2020); Schröder and Biemann (2020) compare MTL with MTL_{All}, and Wang et al. (2018a); Talmor and Berant (2019); Liu et al. (2019b); Phang et al. (2020) use MTL_{All} and STILTs but not pairwise MTL.

In this work we perform comprehensive experiments using all three methods on the 9 datasets in the GLUE benchmark (Wang et al., 2018b). We surprisingly find that a simple size heuristic can be used to determine with more than 92% accuracy which method to use for a given target and supporting task: when the target dataset is larger than the supporting dataset, STILTS should be used; otherwise, MTL should be used (MTL_{All} is almost universally the worst of the methods in our experiments). To confirm the validity of the size heuristic, we additionally perform a targeted experiment varying dataset size for two of the datasets, showing that there is a crossover point in performance between the two methods when the dataset sizes are equal. We believe that this analysis will help NLP researchers to make better decisions when choosing

[1] We make our code publicly available at https://github.com/orionw/MTLvsIFT.

* Corresponding author, oweller2@jhu.edu

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 2: Short Papers, pages 272 - 282

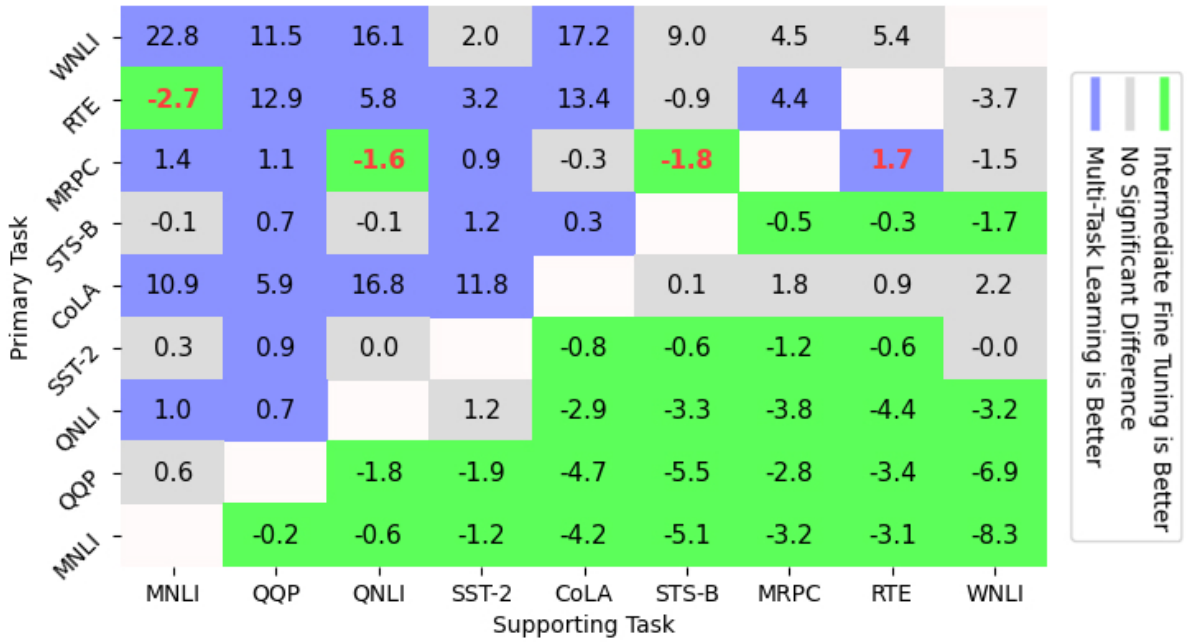

Figure 1: Results comparing intermediate fine tuning (STILTs) vs multi-task learning (MTL). Numbers in cells indicate the absolute percent score difference on the primary task when using MTL instead of STILTs (positive scores mean MTL is better and vice versa). The colors indicate visually the best method, showing a statistically significant difference from the other from using using a two-sided t-test with $\alpha = 0.1$. Numbers in red indicate the cells where the size heuristic does not work. Datasets are ordered in descending size (WNLI is the smallest).

a TL method and will open up future research into understanding the cause of this heuristic's success.

2 Experimental Settings

Dataset Suite To conduct this analysis, we chose to employ the GLUE dataset suite, following and comparing to previous work in transfer learning for NLP (Phang et al., 2018; Liu et al., 2019b).

Training Framework We use Huggingface's *transformers* library (Wolf et al., 2019) for accessing the pre-trained encoder and for the base training framework. We extend this framework to combine multiple tasks into a single PyTorch (Paszke et al., 2017) dataloader for MTL and STILTs training.

Many previous techniques have been proposed for how to best perform MTL (Raffel et al., 2019; Liu et al., 2019b), but a recent paper by Gottumukkala et al. (2020) compared the main approaches and showed that a new dynamic approach provides the best performance in general. We implement all methods described in their paper and experimented with several approaches (sampling by size, uniformity, etc.). Our initial results found that dynamic sampling was indeed the most effective on pairwise tasks. Thus, for the remainder of this paper, our MTL framework uses dynamic sampling with heterogeneous batch schedules. For consistency, we train the STILTs models using the same code, but include only one task in the dataloader instead of multiple. The MTL_{All} setup uses the same MTL code, but includes all 9 GLUE tasks.

We train each model on 5 different seeds to control for randomness (Dodge et al., 2020). For the STILTs method, we train 5 models with different seeds on the supporting task and then choose the best of those models to train with 5 more random seeds on the target task. For our final reported numbers, we record both the average score and the standard deviation, comparing the MTL approach to the STILTs approach with a two-sample t-test. In total, we train $9 * 8 * 5 = 360$ different MTL versions of our model, 5 MTL_{All} models, and $9 * 5 + 9 * 5 = 90$ models in the STILTs setting.

Model We use the DistilRoBERTa model (pretrained and distributed from the *transformers* library similarly to the DistilBERT model in Sanh et al. (2019)) for our experiments, due to its strong performance and efficiency compared to the full model. For details regarding model and compute parameters, see Appendix A. Our purpose is *not* to train the next state-of-the-art model on the GLUE task and thus the absolute scores are not immediately relevant; our purpose is to show how the different methods score *relative to each other*. We note that we conducted the same analysis in Fig-

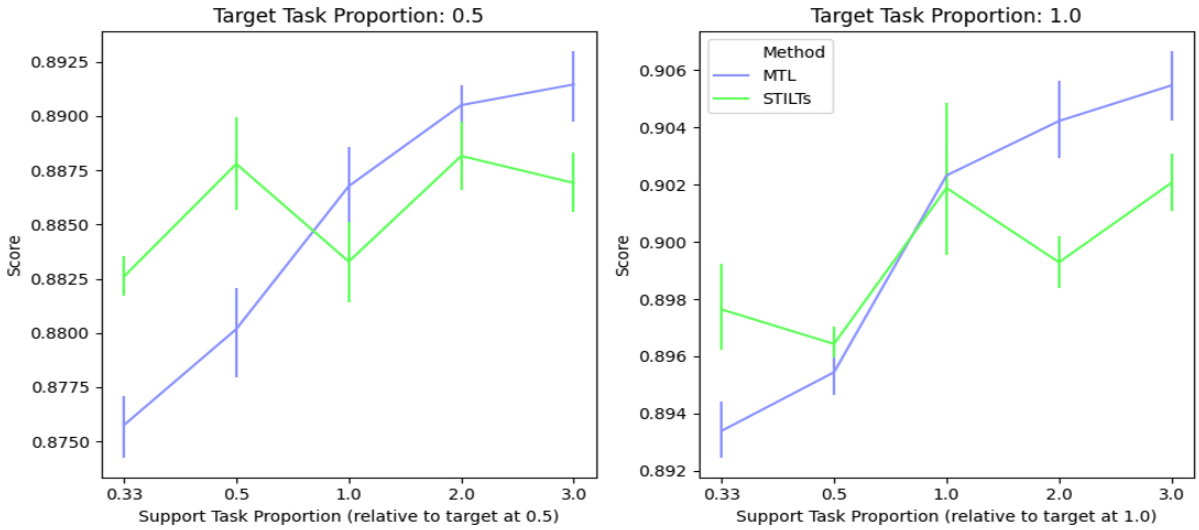

Figure 2: Experiments validating the size heuristic on the (QNLI, MNLI) task pair. The right figure shows training on 100% of the QNLI training set while the left figure shows training with 50%. The x-axis indicates the amount of training data of the supporting task (MNLI) relative to the QNLI training set, artificially constrained (e.g. 0.33 indicates that the supporting task is a third of the size of the QNLI training set, etc.). The blue line indicates MTL results while the green line indicates the STILTs method. Error bars indicate a 90% CI using 5 random seeds.

ure 1 for BERT and found the same conclusion (see Appendix D), showing that our results extend to other pre-trained transformers.

3 Results

We provide three different analyses: a comparison of pairwise MTL vs STILTs, experiments varying dataset size to validate our findings, and a comparison of pairwise approaches vs MTL_{All}.

MTL vs STILTs We first calculate the absolute score matrices from computing the MTL and STILTs method on each pair of the GLUE dataset suite, then subtract the STILTs average score matrix from the MTL one (Figure 1). Thus, this shows the absolute score gain for using the MTL method instead of the STILTs method (negative scores indicate that the STILTs method was better, etc.).

However, this matrix does not tell us whether these differences are statistically significant; for this we use a two-sample t-test to compare the mean and standard deviation of each method for a particular cell. Scores that are statistically significant are color coded green (if STILTs is better) or blue (if MTL is better), whereas they are coded grey if there is no statistically significant difference. We note that although some differences are large (e.g. a 9 point difference on (WNLI, STS-B)) the variance of these results is high enough that there is no statistically significant difference between the STILTs and MTL score distributions.

We order the datasets in Figure 1 by size, to visually illustrate the trend. The number of green cells in a row is highly correlated with the size of the dataset represented by that row. For example, MNLI is the largest and every cell in the MNLI row is green. QQP is the 2nd largest and every cell in its row is also green, except for (QQP, MNLI). The smallest dataset, WNLI, has zero green cells.

We can summarize these results with the following size heuristic: **MTL is better than STILTs when the target task has fewer training instances than the supporting task** and vice versa. In fact, if we use this heuristic to predict which method will be better we find that it predicts 49/53 significant cells, which is equivalent to 92.5% accuracy. To more clearly visualize which cells it fails to predict accurately, those four cells are indicated with red text. We note that this approach does not hold on the cells that have no statistically significant difference between the two methods: but for almost every significant cell, it does.

Unfortunately, there is no clear answer to why those four cells are misclassified. Three of the four misclassified cells come when using the MRPC dataset as the target task, but there is no obvious reason why it fails on MRPC. We recognize that this size heuristic is not an absolute law, but merely a good heuristic that does so with high accuracy: there are still other pieces to this puzzle that this work does not consider, such as dataset similarity.

Dataset Size Experiments In order to validate

Approach	Mean	WNLI	STS-B	SST-2	RTE	QQP	QNLI	MRPC	MNLI	CoLA
MTL_{All}	73.3	54.4	86.6	90.8	**67.4**	80.2	84.9	85.4	74.2	35.8
Avg. STILTs	75.8	45.0	87.5	92.1	61.9	88.9	89.4	**87.4**	**84.0**	46.4
Avg. MTL	77.3	**56.1**	87.4	91.9	66.0	85.6	87.5	**87.4**	80.8	**52.7**
Avg. S.H.	**78.3**	**56.1**	**87.7**	**92.3**	66.5	**89.0**	**89.6**	87.3	**84.0**	52.1
Pairwise Oracle	80.7	57.7	88.8	92.9	76.0	89.5	90.6	90.2	84.3	56.5

Table 1: Comparison of MTL_{All} to the pairwise STILTs or MTL approaches. "S.H" stands for size heuristic. Pairwise Oracle uses the best supplementary task for the given target task using the best pairwise method (STILTs or MTL). All scores are the average of 5 random seeds. We find that on almost every task, pairwise approaches are better than MTL_{All}. Bold scores indicate the best score in the column, excluding the oracle.

the size heuristic further we conduct controlled experiments that alter the amount of training data of the supporting task to be above and below the target task. We choose to test QNLI primary with MNLI supporting, as they should be closely related and thus have the potential to disprove this heuristic. We subsample data from the supporting task so that we have a proportion K of the size of the primary task (where $K \in \{1/3, 1/2, 1, 2, 3\}$). By doing so, we examine whether the size heuristic holds while explicitly controlling for the supporting task's size. Other than dataset size, all experimental parameters are the same as in the original comparison (§2).

We also test whether these results hold if the size of the primary dataset is changed (e.g., perhaps there is something special about the current size of the QNLI dataset). We take the same pair and reduce the training set of QNLI in half, varying MNLI around the new number of instances in the QNLI training set as above (e.g. 1/3rd, 1/2, etc.).

The results of these two experiments are in Figure 2. We can see that as the size of the supporting dataset increases, MTL becomes more effective than STILTs. Furthermore, we find that when both datasets are equal sizes the two methods are statistically similar, as we would expect from the size heuristic (Support Task Proportion=1.0).

Thus, the synthetic experiments corroborate our main finding; the size heuristic holds even on controlled instances where the size of the training sets are artificially manipulated.

Pairwise TL vs MTL_{All} We also experiment with MTL_{All} on GLUE (see Appendix B for implementation details). We find that the average pairwise approach consistently outperforms the MTL_{All} method, except for the RTE task (Table 1) and using the best supporting task outperforms MTL_{All} in every case (Pairwise Oracle). Thus, although MTL_{All} is conceptually simple, it is not the best choice w.r.t. the target task score: on a random dataset simply using STILTs or MTL will likely perform better. Furthermore, using the size heuristic on the average supplementary task increases the score by 5 points over MTL_{All} (78.3 vs 73.3).

4 Related Work

A large body of recent work (Søgaard and Bingel, 2017; Vu et al., 2020; Bettgenhäuser et al., 2020; Peng et al., 2020; Poth et al., 2021) exists that examines *when* these transfer learning methods are more effective than simply fine-tuning on the target task. Oftentimes, these explanations involve recognizing catastrophic forgetting (Phang et al., 2018; Pruksachatkun et al., 2020b; Wang et al., 2018a) although recent work has called for them to be re-examined (Chang and Lu, 2021). This paper is orthogonal to those, as we examine when you should choose MTL or STILTs, rather than when they are more effective than the standard fine-tuning case (in fact, these strategies could be combined to predict transfer and then use the best method). As our task is different, theoretical explanations for how these methods work *in relation to each other* will need to be explored in future work. Potential theories suggested by our results are discussed in Appendix C, and are left to guide those efforts.

5 Conclusion

We examined the three main strategies for transfer learning in natural language processing: training on an intermediate supporting task to aid the target task (STILTs), training on the target and supporting task simultaneously (MTL), or training on multiple supporting tasks alongside the target task (MTL_{All}). We provide the first comprehensive comparison between these three methods using the GLUE dataset suite and show that there is a simple rule for when to use one of these techniques over the other. This simple heuristic, which holds true in more than 92% of applicable cases, states that multi-task learning

is better than intermediate fine tuning when the target task is smaller than the supporting task and vice versa. Additionally, we showed that these pairwise transfer learning techniques outperform the MTL_{All} approach in almost every case.

References

Roy Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second PASCAL recognising textual entailment challenge.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth PASCAL recognizing textual entailment challenge.

Gabriele Bettgenhäuser, Michael A Hedderich, and Dietrich Klakow. 2020. Learning functions to study the benefit of multitask learning. *arXiv preprint arXiv:2006.05561*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Ting-Yun Chang and Chi-Jen Lu. 2021. Rethinking why intermediate-task fine-tuning works. *arXiv preprint arXiv:2108.11696*.

Soravit Changpinyo, Hexiang Hu, and Fei Sha. 2018. Multi-task learning for sequence tagging: An empirical study. *arXiv preprint arXiv:1808.04151*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *ArXiv*, abs/2003.10555.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

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

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the International Workshop on Paraphrasing*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics.

Goran Glavas and I. Vulić. 2020. Is supervised syntactic parsing beneficial for language understanding? an empirical investigation. *ArXiv*, abs/2008.06788.

Ananth Gottumukkala, Dheeru Dua, Sameer Singh, and Matt Gardner. 2020. Dynamic sampling strategies for multi-task reading comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 920–924.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.

N. Keskar, Bryan McCann, Caiming Xiong, and R. Socher. 2019. Unifying question answering and text classification via span extraction. *ArXiv*, abs/1904.09286.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The Winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, volume 46, page 47.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *arXiv preprint arXiv:1904.09482*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019b. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Nicholas Lourie, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Unicorn on rainbow: A universal commonsense reasoning model on a new multitask benchmark. *arXiv preprint arXiv:2103.13009*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Yifan Peng, Qingyu Chen, and Zhiyong Lu. 2020. An empirical study of multi-task learning on bert for biomedical text mining. *arXiv preprint arXiv:2005.02799*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Jason Phang, Phu Mon Htut, Yada Pruksachatkun, Haokun Liu, Clara Vania, Katharina Kann, Iacer Calixto, and Samuel R Bowman. 2020. English intermediate-task training improves zero-shot cross-lingual transfer too. *arXiv preprint arXiv:2005.13013*.

Clifton Poth, Jonas Pfeiffer, Andreas Ruckl'e, and Iryna Gurevych. 2021. What to pre-train on? efficient intermediate task selection. *ArXiv*, abs/2104.08247.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, C. Vania, K. Kann, and Samuel R. Bowman. 2020a. Intermediate-task transfer learning with pretrained models for natural language understanding: When and why does it work? *ArXiv*, abs/2005.00628.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R Bowman. 2020b. Intermediate-task transfer learning with pretrained models for natural language understanding: When and why does it work? *arXiv preprint arXiv:2005.00628*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Fynn Schröder and Chris Biemann. 2020. Estimating the influence of auxiliary tasks for multi-task learning of sequence tagging tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2971–2985.

Damien Sileo, Tim Van-de Cruys, Camille Pradel, and Philippe Muller. 2019. Discourse-based evaluation of language understanding. *arXiv preprint arXiv:1907.08672*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, pages 1631–1642.

Anders Søgaard and Joachim Bingel. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *EACL*.

Alon Talmor and Jonathan Berant. 2019. Multiqa: An empirical investigation of generalization and transfer in reading comprehension. *arXiv preprint arXiv:1905.13453*.

Nikos Voskarides, Dan Li, A. Panteli, and Pengjie Ren. 2019. Ilps at trec 2019 conversational assistant track. In *TREC*.

Tu Vu, Tong Wang, Tsendsuren Munkhdalai, Alessandro Sordoni, Adam Trischler, Andrew Mattarella-Micke, Subhransu Maji, and Mohit Iyyer. 2020. Exploring and predicting transferability across nlp tasks. *arXiv preprint arXiv:2005.00770*.

Alex Wang, Jan Hula, Patrick Xia, Raghavendra Pappagari, R Thomas McCoy, Roma Patel, Najoung Kim, Ian Tenney, Yinghui Huang, Katherin Yu, et al. 2018a. Can you tell me how to get past sesame street? sentence-level pretraining beyond language modeling. *arXiv preprint arXiv:1812.10860*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018b. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2018. Neural network acceptability judgments. *arXiv preprint 1805.12471*.

Orion Weller, Nicholas Lourie, Matt Gardner, and Matthew E. Peters. 2020. Learning from task descriptions. *ArXiv*, abs/2011.08115.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of NAACL-HLT*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yichong Xu, X. Liu, C. Li, Hoifung Poon, and Jianfeng Gao. 2019. Doubletransfer at mediqa 2019: Multi-source transfer learning for natural language understanding in the medical domain. In *BioNLP@ACL*.

Zeyu Yan, Jianqiang Ma, Y. Zhang, and Jianping Shen. 2020. Sql generation via machine reading comprehension. In *COLING*.

Approach	Mean	WNLI	STS-B	SST-2	RTE	QQP	QNLI	MRPC	MNLI	CoLA
MTL_{All} Uniform	63.2	**56.1**	85.1	84.0	58.3	70.4	76.4	80.3	50.7	7.8
MTL_{All} Dynamic	67.2	52.1	86.2	88.4	63.8	75.5	81.2	82.3	64.0	10.9
MTL_{All} Size	**73.3**	54.4	**86.6**	**90.8**	**67.4**	**80.2**	**84.9**	**85.4**	**74.2**	**35.8**
Avg. STILTs	75.8	45.0	87.5	92.1	61.9	88.9	89.4	**87.4**	**84.0**	46.4
Avg. MTL	77.3	**56.1**	87.4	91.9	66.0	85.6	87.5	**87.4**	80.8	**52.7**
Avg. S.H.	**78.3**	**56.1**	**87.7**	**92.3**	66.5	**89.0**	**89.6**	87.3	**84.0**	52.1
Pairwise Oracle	**80.7**	**57.7**	**88.8**	**92.9**	**76.0**	**89.5**	**90.6**	**90.2**	**84.3**	**56.5**

Table 2: Comparison of MTL_{All} to the pairwise STILTs or MTL approaches. "S.H" stands for size heuristic. Pairwise Oracle uses the best supplementary task for the given target task using the best pairwise method (STILTs or MTL). All scores are the average of 5 random seeds. Note that MTL_{All} was run with three different sampling methods (top half). We find that on almost every task, pairwise approaches are better than MTL_{All}. Bold scores indicate the best score in the column for the given section.

Wei Zhu, Xiaofeng Zhou, K. Wang, X. Luo, Xiepeng Li, Y. Ni, and G. Xie. 2019. Panlp at mediqa 2019: Pre-trained language models, transfer learning and knowledge distillation. In *BioNLP@ACL*.

A Training and Compute Details

We use the hyperparameters given by the *transformer* library example on GLUE as the default for our model (learning rate of 2e-5, batch size of 128, AdamW optimizer (Kingma and Ba, 2014), etc.). We train for 10 epochs, checkpointing every half an epoch and use the best model on the development set for the test set scores. We train on a mix of approximately 10 K80 and P100 GPUs for approximately two weeks for the main experiment, using another week of compute time for the synthetic experiments (§3). Our CPUs use 12-core Intel Haswell (2.3 GHz) processors with 32GB of RAM.

B Pairwise Approaches vs MTL_{All}

Experimental Setup We use MTL_{All} with three different sampling methods: uniform sampling, sampling by dataset size, and dynamic sampling. To illustrate the difference between MTL_{All} and the pairwise methods, we show the average score across all supplementary tasks for MTL and STILTs. We also show the average score found by choosing MTL or STILTs using the size heuristic as *Ave. S.H.*. Finally, we report the score from the best task using the best pairwise method, which we call the *Pairwise Oracle*. The results are shown in Table 2.

Results Although dynamic sampling was more effective for the pairwise tasks, we find that dynamic sampling was worse than sampling by size when using MTL on all nine datasets (top half of Table 2).

However, when the MTL_{All} method is compared to the pairwise methods, it does not perform as well (bottom half of Table 2). We see that the Pairwise Oracle, which uses the best supplementary task for the given target task, outperforms all methods by a large margin. Thus, although MTL_{All} is conceptually simple, it is not the best choice with respect to target task accuracy. Furthermore, if you could predict which supplementary task would be most effective (Pairwise Oracle, c.f. Section 4, Vu et al. (2020); Poth et al. (2021), etc.), you would be able to make even larger gains over MTL_{All}.

C Theories for Transfer Effectiveness

Previous work often invokes ideas such as catastrophic forgetting to describe why STILTs or MTL does or does not improve over the basic fine-tuning case (Phang et al., 2018; Pruksachatkun et al., 2020b; Wang et al., 2018a). However, as our work provides a novel comparison of MTL vs. STILTs there exists no previous work that shows how these methods differ in any practical or theoretical terms (e.g. does MTL or STILTs cause more catastrophic forgetting of the target task). Furthermore, previous explanations for why the STILTs method works has been called into question (Chang and Lu, 2021), leaving it an open research area.

A naive explanation for our task would be to think that when the target task is larger, STILTs should be worse because of catastrophic forgetting, whereas MTL would still have access to the supporting task. However, for STILTs this catastrophic forgetting would mainly effect the supporting task performance, not the target task performance, making that explanation unlikely in some contexts (e.g. when the tasks are not closely related). One potential explanation based on our results is that a small supporting task is best used to provide a good ini-

tialization for a larger target task (e.g. STILTs) while a large supporting task used for initialization would change the weights too much for the small target task to use effectively (thus making MTL the more effective strategy for a larger supporting task). Another explanation could be that a larger target task does not benefit from MTL (and perhaps is harmed by it, e.g. catastrophic interference) and therefore, STILTs is more effective - while MTL is more effective for small target tasks. However, all of these explanations also fail to take into account task relatedness, which likely also plays a role in the theoretical explanation (although even that too, has been called into question with Chang and Lu (2021)).

We thus note that there are a myriad of possible explanations (and the answer is likely a complex combination of possible explanations), but these are out of the scope of this work. Our work aims to show what happens in practice, rather than proposing a theoretical framework. As theoretical explanations for transfer learning are still an active area of research, we leave them to future work and provide this empirical comparison to guide their efforts and the current efforts of NLP researchers and practitioners.

D Alternate Model: BERT

We conduct the same analysis as Figure 1 with the BERT model and find similar results (Figure 3, thus showing that our results transfer to other pre-trained transformer models. We follow previous work in using two different pre-trained models for our analysis (Talmor and Berant, 2019; Phang et al., 2018).

E Additional Background Discussion

In this section we will show how the size heuristic is supported by and helps explain the results of previous work in this area. **Although this section is not crucial to the main result of our work, we include it to help readers who may not be as familiar with the related work**. We examine two works in depth and then discuss broader themes of related work.

BERT on STILTs Phang et al. (2018) This work defined the acronym STILTs, or *Supplementary Training on Intermediate Labeled-data Tasks*, which has been an influential idea in the community (Voskarides et al., 2019; Yan et al., 2020; Clark et al., 2020). To determine the effect of the intermediate training, the authors computed the STILTs matrix of each pair in the GLUE dataset. As our model and training framework are different from their methodology, we cannot compare our matrix with the absolute numbers in their matrix. However, at the end of Section 4 in their paper, they conduct an experiment with MTL and compare the results to their STILTs matrix (their experimental results are reproduced in Table 3 for convenience). Their analysis uses MNLI as the supporting task and RTE as the target task, trying MTL, STILTs, MTL+fine-tuning, and only fine-tuning on RTE. Their results show that STILTs provides the highest score, with all MTL varieties being worse. From this they conclude that MTL is worse than STILTs.

Model	RTE accuracy
GPT → RTE	54.2
GPT → MNLI → RTE	**70.4**
GPT → {MNLI, RTE}	68.6
GPT → {MNLI, RTE} → RTE	67.5

Table 3: Table reproduced from Phang et al. (2018). Their comparison of STILTs against MTL setups for GPT, with MNLI as the intermediate task and RTE as the target task. Only one run was reported (e.g. no standard error or confidence intervals).

How does this compare to our results? In Figure 1 we see that our results also show that the STILTs method is better than the MTL method for the (RTE, MNLI) pair, showing that our results are consistent with those in the literature. Furthermore, we find that this is one of the 4 significant cells in our matrix where the size heuristic does not accurately predict the best method. It is unfortunate that the task they decided to pick happened to be one of the anomalies. Thus, our paper extends and completes their results with more rigor.

MultiQA Talmor and Berant (2019) MultiQA showed that using MTL on a variety of question-answering (QA) datasets made it possible to train a model that could outperform the current SOTA on those QA datasets. They used an interesting approach to MTL, pulling 15k examples from each of the 5 major datasets to compose one new "MTL" task, called Multi-75K. They then show results for STILTs transfer on those same datasets along with the MTL dataset (their data is reproduced with new emphasis in Appendix E Table 4 for conve-

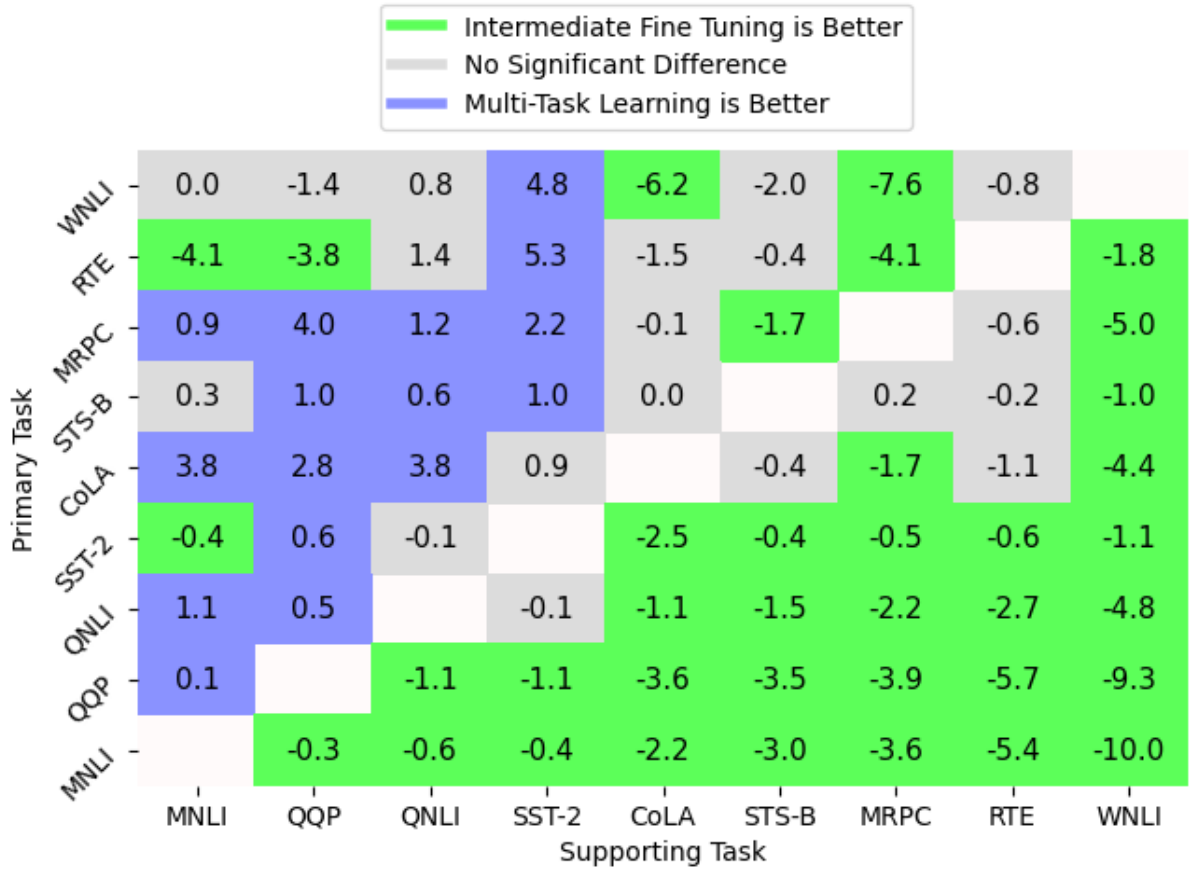

Figure 3: Results comparing intermediate fine tuning (STILTs) vs multi-task learning (MTL) with the BERT model. Numbers in cells indicate the absolute percent score difference on the primary task when using MTL instead of STILTs (positive scores mean MTL is better and vice versa). The colors indicate visually the best method, showing a statistically significant difference from the other from using using a two-sided t-test with $\alpha = 0.1$. Datasets are ordered in descending size.

nience). We note that this STILTs-like transfer with the "MTL" dataset is an equivalent method to doing MTL and then fine-tuning on the target task, reminiscent of the third example in Phang et al. (2018) (Table 3, GPT $\rightarrow$ {MNLI, RTE} $\rightarrow$ RTE, c.f. Appendix E).

How does this relate to our results? The size heuristic says that MTL is better than STILTs when the target task has fewer training instances. In the MultiQA paper the size of each training set is artificially controlled to be the same number (75k instances), thus our size heuristic would say that the methods should be comparable. Although no error bounds or standard deviations are reported in their paper (which makes the exact comparison difficult), we see that the MTL approach performs equal or better on almost half of the datasets. Thus, although the MultiQA paper is not strictly comparable to our work due to their training setup (the MTL+fine tuning), their results agree with our hypothesis as well.

For convenience, Table 4 from Talmor and Berant (2019) is reproduced here in the appendix. The top half contains the results using the DocQA model while the bottom half uses BERT. Note that both model's Multi-75K scores perform approximately similar to the STILTs methods, which is expected given that they are the same size. TQA-G and TQA-W come from the same dataset. As stated in the body of this paper, no standard deviation is reported in the MultiQA paper and thus it is hard to know whether the difference in results are statistically significant. Even if all results were statistically significant, which is highly unlikely, each of the Multi-75K models perform equal or better on 2 of the 6 tasks, which is not statistically different from random.

Combining All Tasks Our results using MTL_{All} showed that although MTL_{All} is conceptually easy (just put all the datasets together) it does not lead to the best performance. We find similar results in Wang et al. (2018a), where in their Table 3 they show that the STILTs approach outperforms the

	SQuAD	NewsQA	SearchQA	TQA-G	TQA-W	HotpotQA
SQuAD	-	**33.3**	39.2	49.2	34.5	17.8
NewsQA	59.6	-	41.6	44.2	33.9	16.5
SearchQA	57	31.4	-	**57.5**	39.6	**19.2**
TQA-G	57.7	31.8	**49.5**	-	**41.4**	19.1
TQA-W	57.6	31.7	44.4	50.7	-	17.2
HotpotQA	**59.8**	32.4	46.3	54.6	37.4	-
Multi-75K	**59.8**	33.0	47.5	56.4	40.4	**19.2**
SQuAD	-	41.2	47.8	55.2	45.4	20.8
NewsQA	**72.1**	-	47.4	55.9	45.2	20.6
SearchQA	70.2	40.2	-	**57.3**	45.5	20.4
TQA-G	69.9	41.2	**50.0**	-	46.2	20.8
TQA-W	71.0	39.2	48.4	55.7	-	**20.9**
HotpotQA	71.2	39.5	48.6	56.6	45.6	-
Multi-75K	71.5	**42.1**	48.5	56.6	**46.5**	20.4

Table 4: Results taken from the right half of Table 4 in the MultiQA paper (Talmor and Berant, 2019) as that section is directly relevant to this work (the *self* row containing only standard fine-tuning is removed for clarity). Emphasis changed to reflect the best score in the model's column instead of the best non-MTL score.

MTL_{All} approach for all but one task. Additionally, in the follow up work from the initial STILTs paper (Phang et al., 2020) they find that although MTL_{All} has a slightly higher average performance in the cross-lingual setting, it is worse than the pairwise approach in 75% of the evaluated tasks.

The current literature (and our work) seems to suggest that naively combining as many tasks as possible may not be the best approach. However, more work is needed to understand the training dynamics of MTL_{All}.

Combining Helpful Tasks In this paper, we only examine the difference between pairwise MTL, STILTs or MTL_{All}, due to time and space. Although it is possible that our heuristic may extrapolate to transfer learning with more than two tasks, computing the power set of the possible task combinations for MTL and STILTs would be extremely time and resource intensive. We leave it to future work to examine how the size heuristic may hold when using more than two datasets at a time.

Additionally, there may be further value in computing this power set: Changpinyo et al. (2018) showed that taking the pairwise tasks that proved beneficial in pairwise MTL and combining them into a larger MTL set (an "Oracle" set) oftentimes provides higher scores than pairwise MTL. Exploring which subsets of tasks provide the best transfer with which method would be valuable future work.

Dataset Size in TL Dataset size has been used often in transfer learning techniques (Søgaard and Bingel, 2017; Pruksachatkun et al., 2020a; Poth et al., 2021). Our size heuristic, although related, focuses on a different problem: whether to use MTL or STILTs. Thus, our work provides additional insight into how the size of the dataset is important for transfer learning.

Fine-tuning after MTL Many papers that use MTL_{All} also perform some sort of fine-tuning after the MTL phase. Since fine-tuning after MTL makes the MTL phase an intermediate step, it essential combines the STILTs and MTL methods into a single STILTs-like method. However, whether fine-tuning after MTL is better than simply MTL is still controversial: for example, Liu et al. (2019b), Raffel et al. (2019), and Talmor and Berant (2019) say that fine-tuning after MTL helps but Lourie et al. (2021) and Phang et al. (2018) say that it doesn't. However, Raffel et al. (2019) is the only one whose experiments include multiple random seeds, giving more credence to their results. However, due to the difference of opinion it is unclear which method is actually better; we leave this to future work.

F GLUE Dataset Sizes and References

To give credit to the original authors and to provide the exact sizes, we provide Table 5.

Dataset	Citation	Training Size
MNLI	Williams et al. (2018)	392,662
QQP	No citation, link here	363,846
QNLI	Levesque et al. (2011)	104,743
SST-2	Socher et al. (2013)	67,349
CoLA	Warstadt et al. (2018)	8,551
STS-B	Cer et al. (2017)	5,749
MRPC	Dolan and Brockett (2005)	3,668
RTE	Dagan et al. (2006)*	2,490
WNLI	Levesque et al. (2011)	635

Table 5: Sizes of the datasets in GLUE (Wang et al., 2018b) in descending order, along with their original citations. RTE is compiled from these sources: Dagan et al. (2006); Bar Haim et al. (2006); Giampiccolo et al. (2007); Bentivogli et al. (2009)