BAM! Born-Again Multi-Task Networks for Natural Language Understanding

Kevin Clark[†] **Minh-Thang Luong**[‡] **Urvashi Khandelwal**[†]
Christopher D. Manning[†] **Quoc V. Le**[‡]
[†]Computer Science Department, Stanford University
[‡]Google Brain
{kevclark,urvashik,manning}@cs.stanford.edu
{thangluong,qvl}@google.com

Abstract

It can be challenging to train multi-task neural networks that outperform or even match their single-task counterparts. To help address this, we propose using knowledge distillation where single-task models teach a multi-task model. We enhance this training with teacher annealing, a novel method that gradually transitions the model from distillation to supervised learning, helping the multi-task model surpass its single-task teachers. We evaluate our approach by multi-task fine-tuning BERT on the GLUE benchmark. Our method consistently improves over standard single-task and multi-task training.

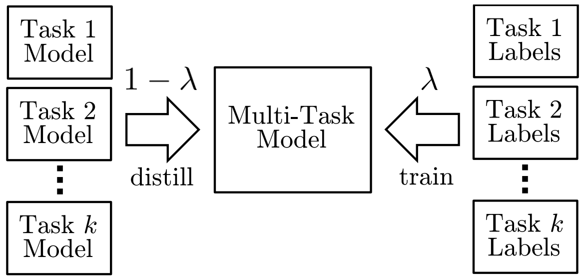

Figure 1: An overview of our method. λ is increased linearly from 0 to 1 over the course of training.

1 Introduction

Building a single model that jointly learns to perform many tasks effectively has been a long-standing challenge in Natural Language Processing (NLP). However, applying multi-task NLP remains difficult for many applications, with multi-task models often performing worse than their single-task counterparts (Plank and Alonso, 2017; Bingel and Søgaard, 2017; McCann et al., 2018). Motivated by these results, we propose a way of applying knowledge distillation (Bucilu et al., 2006; Ba and Caruana, 2014; Hinton et al., 2015) so that single-task models effectively teach a multi-task model.

Knowledge distillation transfers knowledge from a "teacher" model to a "student" model by training the student to imitate the teacher's outputs. In "born-again networks" (Furlanello et al., 2018), the teacher and student have the same neural architecture and model size, but surprisingly the student is able to surpass the teacher's accuracy. Intuitively, distillation is effective because the teacher's output distribution over classes provides more training signal than a one-hot label; Hinton et al. (2015) suggest that teacher outputs contain "dark knowledge" capturing additional information about training examples.

Our work extends born-again networks to the multi-task setting. We compare Single→Multi[1] born-again distillation with several other variants (Single→Single and Multi→Multi), and also explore performing multiple rounds of distillation (Single→Multi→Single→Multi). Furthermore, we propose a simple teacher annealing method that helps the student model outperform its teachers. Teacher annealing gradually transitions the student from learning from the teacher to learning from the gold labels. This method ensures the student gets a rich training signal early in training but is not limited to only imitating the teacher.

Our experiments build upon recent success in self-supervised pre-training (Dai and Le, 2015; Peters et al., 2018) and multi-task fine-tune BERT (Devlin et al., 2019) to perform the tasks from the GLUE natural language understanding benchmark (Wang et al., 2019). Our training method, which we call Born-Again Multi-tasking (BAM)[2], consistently outperforms standard single-task and multi-task training. Further analysis shows the multi-task models benefit from both better regu-

[1]We use Single→Multi to indicate distilling single-task "teacher" models into a multi-task "student" model.

[2]Code is available at https://github.com/google-research/google-research/tree/master/bam

larization and transfer between related tasks.

2 Related Work

Multi-task learning for neural networks in general (Caruana, 1997) and within NLP specifically (Collobert and Weston, 2008; Luong et al., 2016) has been widely studied. Much of the recent work for NLP has centered on neural architecture design: e.g., ensuring only beneficial information is shared across tasks (Liu et al., 2017; Ruder et al., 2019) or arranging tasks in linguistically-motivated hierarchies (Søgaard and Goldberg, 2016; Hashimoto et al., 2017; Sanh et al., 2019). These contributions are orthogonal to ours because we instead focus on the multi-task training algorithm.

Distilling large models into small models (Kim and Rush, 2016; Mou et al., 2016) or ensembles of models into single models (Kuncoro et al., 2016; Liu et al., 2019a) has been shown to improve results for many NLP tasks. There has also been some work on using knowledge distillation to aide in multi-task learning. In reinforcement learning, knowledge distillation has been used to regularize multi-task agents (Parisotto et al., 2016; Teh et al., 2017). In NLP, Tan et al. (2019) distill single-language-pair machine translation systems into a many-language system. However, they focus on multilingual rather than multi-task learning, use a more complex training procedure, and only experiment with Single→Multi distillation.

Concurrently with our work, several other recent works also explore fine-tuning BERT using multiple tasks (Phang et al., 2018; Liu et al., 2019b; Keskar et al., 2019). However, they use only standard transfer or multi-task learning, instead focusing on finding beneficial task pairs or designing improved task-specific components on top of BERT.

3 Methods

3.1 Multi-Task Setup

Model. All of our models are built on top of BERT (Devlin et al., 2019). This model passes byte-pair-tokenized (Sennrich et al., 2016) input sentences through a Transformer network (Vaswani et al., 2017), producing a contextualized representation for each token. The vector corresponding to the first input token[3] c is passed into a task-specific classifier. For classification tasks, we use a standard softmax layer: softmax(Wc). For regression tasks, we normalize the labels so they are between 0 and 1 and then use a size-1 NN layer with a sigmoid activation: sigmoid$(w^T c)$. In our multi-task models, all of the model parameters are shared across tasks except for these classifiers on top of BERT, which means less than 0.01% of the parameters are task-specific. Following BERT, the token embeddings and Transformer are initialized with weights from a self-supervised pre-training phase.

Training. Single-task training is performed as in Devlin et al. (2019). For multi-task training, examples of different tasks are shuffled together, even within minibatches. The summed loss across all tasks is minimized.

3.2 Knowledge Distillation

We use $\mathcal{D}_\tau = \{(x_\tau^1, y_\tau^1), ..., (x_\tau^N, y_\tau^N)\}$ to denote the training set for a task τ and $f_\tau(x, \theta)$ to denote the outputs for task τ produced by a neural network with parameters θ on the input x (for classification tasks this is a distribution over classes). Standard supervised learning trains θ to minimize the loss on the training set:

$$\mathcal{L}(\theta) = \sum_{x_\tau^i, y_\tau^i \in \mathcal{D}_\tau} \ell(y_\tau^i, f_\tau(x_\tau^i, \theta))$$

where for classification tasks ℓ is usually cross-entropy. Knowledge distillation trains the model to instead match the predictions of a teacher model with parameters θ':

$$\mathcal{L}(\theta) = \sum_{x_\tau^i, y_\tau^i \in \mathcal{D}_\tau} \ell(f_\tau(x_\tau^i, \theta'), f_\tau(x_\tau^i, \theta))$$

Note that our distilled networks are "born-again" in that the student has the same model architecture as the teacher, i.e., all of our models have the same prediction function f_τ for each task. For regression tasks, we train the student to minimize the L2 distance between its prediction and the teacher's instead of using cross-entropy loss. Intuitively, knowledge distillation improves training because the full distribution over labels provided by the teacher provides a richer training signal than a one-hot label. See Furlanello et al. (2018) for a more thorough discussion.

Multi-Task Distillation. Given a set of tasks $\mathcal{T}$, we train a single-task model with parameters θ_τ on each task τ. For most experiments, we use

[3]For BERT this is a special token [CLS] that is prepended to each input sequence.

the single-task models to teach a multi-task model with parameters θ:

$$\mathcal{L}(\theta) = \sum_{\tau \in \mathcal{T}} \sum_{x_\tau^i, y_\tau^i \in \mathcal{D}_\tau} \ell(f_\tau(x_\tau^i, \theta_\tau), f_\tau(x_\tau^i, \theta))$$

However, we experiment with other distillation strategies as well.

Teacher Annealing. In knowledge distillation, the student is trained to imitate the teacher. This raises the concern that the student may be limited by the teacher's performance and not be able to substantially outperform the teacher. To address this, we propose *teacher annealing*, which mixes the teacher prediction with the gold label during training. Specifically, the term in the summation becomes

$$\ell(\lambda y_\tau^i + (1 - \lambda) f_\tau(x_\tau^i, \theta_\tau), f_\tau(x_\tau^i, \theta))$$

where λ is linearly increased from 0 to 1 throughout training. Early in training, the model is mostly distilling to get as useful of a training signal as possible. Towards the end of training, the model is mostly relying on the gold-standard labels so it can learn to surpass its teachers.

4 Experiments

Data. We use the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019), which consists of 9 natural language understanding tasks on English data. Tasks cover textual entailment (RTE and MNLI) question-answer entailment (QNLI), paraphrase (MRPC), question paraphrase (QQP), textual similarity (STS), sentiment (SST-2), linguistic acceptability (CoLA), and Winograd Schema (WNLI).

Training Details. Rather than simply shuffling the datasets for our multi-task models, we follow the task sampling procedure from Bowman et al. (2018), where the probability of training on an example for a particular task τ is proportional to $|\mathcal{D}_\tau|^{0.75}$. This ensures that tasks with very large datasets don't overly dominate the training.

We also use the layerwise-learning-rate trick from Howard and Ruder (2018). If layer 0 is the NN layer closest to the output, the learning rate for a particular layer d is set to $\text{BASE\_LR} \cdot \alpha^d$ (i.e., layers closest to the input get lower learning rates). The intuition is that pre-trained layers closer to the input learn more general features, so they shouldn't be altered much during training.

Hyperparameters. For single-task models, we use the same hyperparameters as in the original BERT experiments except we pick a layerwise-learning-rate decay α of 1.0 or 0.9 on the dev set for each task. For multi-task models, we train the model for longer (6 epochs instead of 3) and with a larger batch size (128 instead of 32), using $\alpha = 0.9$ and a learning rate of 1e-4. All models use the BERT-Large pre-trained weights.

Reporting Results. Dev set results report the average score (Spearman correlation for STS, Matthews correlation for CoLA, and accuracy for the other tasks) on all GLUE tasks except WNLI, for which methods can't outperform a majority baseline. Results show the median score of at least 20 trials with different random seeds. We find using a large number of trials is essential because results can vary significantly for different runs. For example, standard deviations in score are over ± 1 for CoLA, RTE, and MRPC for multi-task models. Single-task standard deviations are even larger.

5 Results

Main Results. We compare models trained with single-task learning, multi-task learning, and several varieties of distillation in Table 1. While standard multi-task training improves over single-task training for RTE (likely because it is closely related to MNLI), there is no improvement on the other tasks. In contrast, Single→Multi knowledge distillation improves or matches the performance of the other methods on all tasks except STS, the only regression task in GLUE. We believe distillation does not work well for regression tasks because there is no distribution over classes passed on by the teacher to aid learning.

The gain for Single→Multi over Multi is larger than the gain for Single→Single over Single, suggesting that distillation works particularly well in combination with multi-task learning. Interestingly, Single→Multi works substantially better than Multi→Multi distillation. We speculate it may help that the student is exposed to a diverse set of teachers in the same way ensembles benefit from a diverse set of models, but future work is required to fully understand this phenomenon. In addition to the models reported in the table, we also trained Single→Multi→Single→Multi models. However, the difference with Single→Multi was not statistically significant, suggesting there is little value in multiple rounds of distillation.

Model	Avg.	CoLA[a] $\lvert\mathcal{D}\rvert$ = 8.5k	SST-2[b] 67k	MRPC[c] 3.7k	STS-B[d] 5.8k	QQP[e] 364k	MNLI[f] 393k	QNLI[g] 108k	RTE[h] 2.5k
Single	84.0	60.6	93.2	88.0	90.0	91.3	86.6	92.3	70.4
Multi	85.5	60.3	93.3	88.0	89.8	91.4	86.5	92.2	82.1
Single→Single	84.3	**61.7****	93.2	**88.7***	90.0	91.4	**86.8****	**92.5*****	70.0
Multi→Multi	85.6	60.9	93.5	88.1	89.8	**91.5***	86.7	92.3	82.0
Single→Multi	**86.0*****	**61.8****	**93.6***	**89.3****	89.7	**91.6***	**87.0*****	**92.5*****	**82.8***

Dataset references: [a]Warstadt et al. (2018) [b]Socher et al. (2013) [c]Dolan and Brockett (2005) [d]Cer et al. (2017) [e]Iyer et al. (2017) [f]Williams et al. (2018) [g]constructed from SQuAD (Rajpurkar et al., 2016) [h]Giampiccolo et al. (2007)

Table 1: Comparison of methods on the GLUE dev set. *, **, and *** indicate statistically significant ($p < .05$, $p < .01$, and $p < .001$) improvements over both Single and Multi according to bootstrap hypothesis tests.[3]

Model	GLUE score
BERT-Base (Devlin et al., 2019)	78.5
BERT-Large (Devlin et al., 2019)	80.5
BERT on STILTs (Phang et al., 2018)	82.0
MT-DNN (Liu et al., 2019b)	82.2
Span-Extractive BERT on STILTs (Keskar et al., 2019)	82.3
Snorkel MeTaL ensemble (Hancock et al., 2019)	83.2
MT-DNN$_{KD}$* (Liu et al., 2019a)	83.7
BERT-Large + BAM **(ours)**	82.3

Table 2: Comparison of test set results. *MT-DNN$_{KD}$ is distilled from a diverse ensemble of models.

Overall, a key benefit of our method is robustness: while standard multi-task learning produces mixed results, Single→Multi distillation consistently outperforms standard single-task and multi-task training. We also note that in some trials single-task training resulted in models that score quite poorly (e.g., less than 91 for QQP or less than 70 for MRPC), while the multi-task models have more dependable performance.

Test Set Results. We compare against recent work by submitting to the GLUE leaderboard. We use Single→Multi distillation. Following the procedure used by BERT, we train multiple models and submit the one with the highest average dev set score to the test set. BERT trained 10 models for each task (80 total); we trained 20 multi-task models. Results are shown in Table 2.

Our work outperforms or matches existing published results that do not rely on ensembling. However, due to the variance between trials discussed under "Reporting Results," we think these test set numbers should be taken with a grain of salt, as they only show the performance of individual training runs (which is further complicated by the use of tricks such as dev set model selection). We believe significance testing over multiple trials would be needed to have a definitive comparison.

Single-Task Fine-Tuning. A crucial difference distinguishing our work from the STILTs, Snorkel MeTaL, and MT-DNN$_{KD}$ methods in Table 2 is that we do not single-task fine-tune our model. That is, we do not further train the model on individual tasks after the multi-task training finishes. While single-task fine-tuning improves results, we think to some extent it defeats the purpose of multi-task learning: the result of training is one model for each task instead of a model that can perform all of the tasks. Compared to having many single-task models, a multi-task model is simpler to deploy, faster to run, and arguably more scientifically interesting from the perspective of building general language-processing systems.

We evaluate the benefits of single-task fine-tuning in Table 3. Single-task fine-tuning initializes models with multi-task-learned weights and then performs single-task training. Hyperparameters are the same as for our single-task models except we use a smaller learning rate of 1e-5. While single-task fine-tuning unsurprisingly improves results, the gain on top of Single→Multi distillation is small, reinforcing the claim that distillation provides many of the benefits of single-task training while producing a single unified model instead of many task-specific models.

Ablation Study. We show the importance of teacher annealing and the other training tricks in Table 4. We found them all to significantly

[3]For all statistical tests we use the Holm-Bonferroni method (Holm, 1979) to correct for multiple comparisons.

Model	Avg. Score
Multi	85.5
+Single-Task Fine-Tuning	+0.3
Single→Multi	86.0
+Single-Task Fine-Tuning	+0.1

Table 3: Combining multi-task training with single-task fine-tuning. Improvements are statistically significant ($p < .01$) according to Mann-Whitney U tests.[3]

Model	Avg. Score
Single→Multi	86.0
No layer-wise LRs	−0.3
No task sampling	−0.4
No teacher annealing: $\lambda = 0$	−0.5
No teacher annealing: $\lambda = 0.5$	−0.3

Table 4: Ablation Study. Differences from Single→Multi are statistically significant ($p < .001$) according to Mann-Whitney U tests.[3]

improve scores. Using pure distillation without teacher annealing (i.e., fixing $\lambda = 0$) performs no better than standard multi-task learning, demonstrating the importance of the proposed teacher annealing method.

Comparing Combinations of Tasks. Training on a large number of tasks is known to help regularize multi-task models (Ruder, 2017). A related benefit of multi-task learning is the transfer of learned "knowledge" between closely related tasks. We investigate these two benefits by comparing several models on the RTE task, including one trained with a very closely related task (MNLI, a much large textual entailment dataset) and one trained with fairly unrelated tasks (QQP, CoLA, and SST). We use Single→Multi distillation (Single→Single in the case of the RTE-only model). Results are shown in Table 5. We find both sets of auxiliary tasks improve RTE performance, suggesting that both benefits are playing a role in improving multi-task models. Interestingly, RTE + MNLI alone slightly outperforms the model performing all tasks, perhaps because training on MNLI, which has a very large dataset, is already enough to sufficiently regularize the model.

6 Discussion and Conclusion

We have shown that Single→Multi distillation combined with teacher annealing produces results consistently better than standard single-task or multi-task training. Achieving robust multi-task gains across many tasks has remained elusive in previous research, so we hope our work will make multi-task learning more broadly useful within NLP. However, with the exception of closely related tasks with small datasets (e.g., MNLI helping RTE), the overall size of the gains from our multi-task method are small compared to the gains provided by transfer learning from self-supervised tasks (i.e., BERT). It remains to be fully understood to what extent "self-supervised pre-training is all you need" and where transfer/multi-task learning from supervised tasks can provide the most value.

Trained Tasks	RTE score
RTE	70.0
RTE + MNLI	83.4
RTE + QQP + CoLA + SST	75.1
All GLUE	82.8

Table 5: Which tasks help RTE? Pairwise differences are statistically significant ($p < .01$) according to Mann-Whitney U tests.[3]

Acknowledgements

We thank Robin Jia, John Hewitt, and the anonymous reviewers for their thoughtful comments and suggestions. Kevin is supported by a Google PhD Fellowship.

References

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In *NIPS*.

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

Samuel R Bowman, Ellie Pavlick, Edouard Grave, Benjamin Van Durme, Alex Wang, Jan Hula, Patrick Xia, Raghavendra Pappagari, R Thomas McCoy, Roma Patel, et al. 2018. Looking for ELMo's friends: Sentence-level pretraining beyond language modeling. *arXiv preprint arXiv:1812.10860*.

Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *SIGKDD*.

Rich Caruana. 1997. Multitask learning. *Machine Learning*.

Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *SemEval@ACL*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *NIPS*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *IWP@IJCNLP*.

Tommaso Furlanello, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born again neural networks. In *ICML*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B. Dolan. 2007. The third pascal recognizing textual entailment challenge. In *ACL-PASCAL@ACL*.

Braden Hancock, Clara McCreery, Ines Chami, Vincent Chen, Sen Wu, Jared Dunnmon, Paroma Varma, Max Lam, and Chris R. 2019. Massive multi-task learning with snorkel metal: Bringing more supervision to bear.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *EMNLP*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Shankar Iyer, Nikhil Dandekar, and Kornl Csernai. 2017. First quora dataset release: Question pairs.

Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Unifying question answering and text classification via span extraction. *arXiv preprint arXiv:1904.09286*.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *EMNLP*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an ensemble of greedy dependency parsers into one mst parser. In *EMNLP*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *ACL*.

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*.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *ICLR*.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Lili Mou, Ran Jia, Yan Xu, Ge Li, Lu Zhang, and Zhi Jin. 2016. Distilling word embeddings: An encoding approach. In *CIKM*.

Emilio Parisotto, Jimmy Ba, and Ruslan Salakhutdinov. 2016. Actor-mimic: Deep multitask and transfer reinforcement learning. In *ICLR*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

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*.

Barbara Plank and Héctor Martínez Alonso. 2017. When is multitask learning effective? Semantic sequence prediction under varying data conditions. In *EACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy S. Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2019. Latent multi-task architecture learning. In *AAAI*.

Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2019. A hierarchical multi-task approach for learning embeddings from semantic tasks. In *AAAI*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

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

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL*.

Xu Tan, Yi Ren, Di He, Tao Qin, and Tie-Yan Liu. 2019. Multilingual neural machine translation with knowledge distillation. In *ICLR*.

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. 2017. Distral: Robust multitask reinforcement learning. In *NIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

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

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