AN OVERVIEW OF MODELS AND METHODS FOR MULTI-TASK LEARNING

Wenxuan Zhou

University of Southern California
zhouwenx@usc.edu

October 8, 2019

ABSTRACT

Multi-task learning (MTL) has led to successes in many applications of machine learning, including natural language processing, speech recognition and computer vision. It aims to leverage useful information from multiple related tasks to help improve the generalization performance of all the tasks. In this overview, we first introduce its motivation and applications in NLP, then discuss the challenges of applying MTL and corresponding solutions.

1 Introduction to Multi-Task Learning

Deep Learning has achieved state-of-the-art performance on a wide range of tasks. In deep learning, we typically care about optimizing for a particular metric. In order to do this, we generally train a single model or an ensemble of models to perform our desired task. We then fine-tune and tweak these models until their performance no longer increases. While we can generally achieve acceptable performance this way, by being laser-focused on our single task, we ignore information that might help us do even better on the metric we care about. Specifically, in machine learning we have multiple tasks, where all of them or at least a subset of them are assumed to be related to each other. By sharing representations between related tasks, we can enable our model to generalize better on our original task. This approach is called Multi-task Learning (MTL), which can be formally defined by:

Definition 1.1. (**Multi-task learning** [1]) Given m learning tasks $\{T_i\}_{i=1}^m$ where all the tasks or a subset of them are related, multi-task learning aims to help improve the learning of a model for T_i by using the knowledge contained in all or some of the m tasks.

Motivation. Multi-task learning is inspired by human learning activities where people often apply the knowledge learned from previous tasks to help learn a new task. For example, for a person who learns to ride the bicycle and tricycle together, the experience in learning to ride a bicycle can be utilized in riding a tricycle and vice versa. Similar to human learning, it is useful for multiple learning tasks to be learned jointly since the knowledge contained in a task can be leveraged by other tasks. To better understand multi-task learning in a machine learning perspective, we introduce the mechanisms under multi-task learning, most of which are covered by [2]. For all examples, we will assume that we have two related tasks A and B, which rely on a common hidden layer representation F.

- **Implicit data augmentation.** MTL effectively increases the sample size that we are using for training our model. As all tasks are at least somewhat noisy, when training a model on some task A, our aim is to learn a good representation for task A that ideally ignores the data-dependent noise and generalizes well. As different tasks have different noise patterns, a model that learns two tasks simultaneously is able to learn a more general representation. Learning just task A bears the risk of overfitting to task A, while learning A and B jointly enables the model to obtain a better representation F through averaging the noise patterns.
- **Representation bias.** MTL biases the model to prefer representations that other tasks also prefer. This will also help the model to generalize to new tasks in the future as a hypothesis space that performs well for a

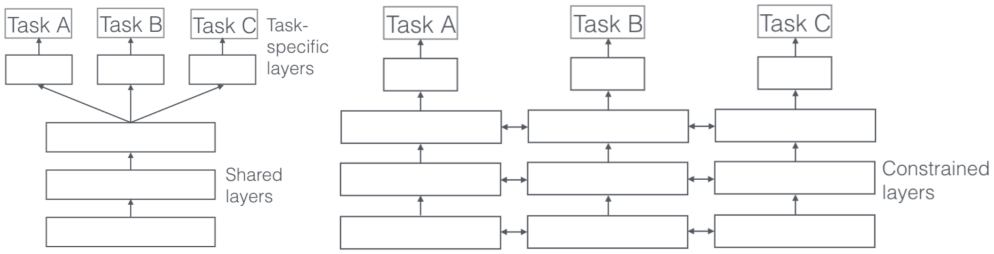


Figure 1: Hard parameter sharing (left) and soft parameter sharing (right) in deep multi-task learning.

sufficiently large number of training tasks will also perform well for learning novel tasks as long as they are from the same environment.

- **Eavesdropping.** Some features G are easy to learn for some task B, while being difficult to learn for another task A. This might either be because A interacts with the features in a more complex way or because other features are impeding the model's ability to learn G. Through MTL, we can allow the model to eavesdrop. Related work in deep learning includes supervision from low-level tasks [3].

Methods. In multi-tasking learning, we aim to jointly train models $f_i(x; \theta_i)$ for each task $T_i = (\mathcal{X}_i, \mathcal{Y}_i)$, which yields the following loss function:

$$\min_{\{\theta_i\}_{i=1}^m} \sum_{i=1}^{m} a_i \mathcal{L}_i(\mathcal{X}_i, \mathcal{Y}_i; \theta_i) \tag{1}$$

where a_i is the weight for task T_i. In order to utilize knowledge from other tasks, we may want to share their neural structures and model parameters. The two most common sharing mechanisms are hard parameter sharing and soft parameter sharing, as shown in Figure 1. *Hard parameter sharing* is the most commonly used approach to MTL in neural networks. It is generally applied by stacking task-specific classifiers on a shared representation. Hard parameter sharing greatly reduces the risk of overfitting. In fact, [4] showed that the risk of overfitting the shared parameters is an $O(N)$ – where N is the number of tasks – smaller than overfitting the task-specific parameters, i.e. the output layers. This makes sense intuitively: The more tasks we are learning simultaneously, the more our model has to find a representation that captures all of the tasks and the less is our chance of overfitting on our original task. While in *Soft parameter sharing*, each model has its own parameters, but the distances between the model parameters are encouraged to be similar. Besides these two methods, many works adopt a hybrid way – they learn a shared representation as well as task-specific features.

Key Challenges. Multi-task learning has been applied on a wide range of NLP tasks, such as representation learning [5], machine translation [6, 7], text classification [8, 9, 10] and sequence labeling [3, 10]. However, multi-task learning proposes the following challenges:

- **Task / Dataset Selection.** Multi-task learning works by sharing the knowledge of related tasks. However, it is hard to decide what knowledge can be shared. In current works, task selection is mostly motivated from traditional methods or common sense. But it does not always work. For example, [11] uses semantic role labeling [1] as an auxiliary task to improve the relation extraction model but shows no improvement.
- **Destructive Interference [12]** In multi-task learning, some tasks may be weakly-related, they provide competing or even contradicting gradient directions during training. For example, words "excellent" and "bad" are considered similar in syntactic tasks (part-of-speech, dependency parsing), but have opposite meanings in sentiment analysis. When these weakly-related tasks are jointly trained, their gradients will be cancelled out, which leads the model stuck in local minima.
- **Task Weighing.** As shown Equation 2, losses from different tasks are unequally weighted. It is important for learning task-specific classifiers because of the difference in task difficulty and scale of loss functions. Most works determine the weight by grid searching, which is infeasible for large number of tasks.

[1] A task to determine the semantic role of each work in a sentence, such as agent, goal or result.

2 Applications in NLP

In this section, we introduce some recent advances of multi-task learning in NLP. We focus on general models and methods instead of task details.

2.1 JMT [3]

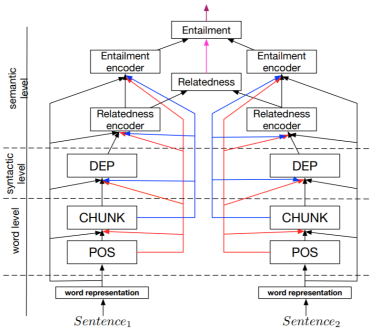


Figure 2: The architecture of Joint Many-Task (JMT) model.

The Joint Many-Task (JMT) model introduces the linguistic hierarchy, where syntactic tasks (POS tagging, chunking, dependency parsing) are predicted in low-level layers while semantic tasks (text relatedness, entailment) are predicted in high-level layers. They jointly train these tasks and achieve universal improvements. Besides parameter sharing, the low-level tasks also provide additional features for high-level tasks. Specifically, given a sentence x and pretrained word embeddings $\{e_t\}_{t=1}^{L}$, JMT first transforms the sentence into contextual embeddings $\{h_t^{(1)}\}_{t=1}^{L}$ by bidirectional LSTM, then predict the POS tags by:

$$P(y_t^{(1)} = j|x) = \text{softmax}(W_1 h_t^{(1)})$$

For the chunking task, JMT stacks a bi-LSTM layer upon POS model to get the task-specific embedding $\{h_t^{(2)}\}_{t=1}^{L}$. To better utilize the knowledge of previous tasks, the chunking model also takes the weighted POS embedding as input, which is calculated by:

$$y_t^{(pos)} = \sum_{j=1}^{C} P(y_t^{(1)} = j|x) l(j)$$

Then the input representation for chunking classifier is $[h_{t-1}^{(2)}, h_t^{(1)}, e_t, y_t^{(pos)}]$. Other tasks are predicted in similar ways.

In each training epoch, JMT trains low-level tasks first finetune the layers in high-level tasks. In order to prevent catastrophic forgetting, it proposes *successive regularization*, which requires the low-level parameters do not change too much in high-level finetuning. Specifically, it is calculated by $||\theta - \theta'||_2^2$, where θ / θ' denote the model parameters before / after finetuning.

2.2 LISA [9]

Linguistically-Informed Self-Attention (LISA) combines multi-head self attention with multi-task learning across POS tagging, dependency parsing, predicate detection and semantic role labeling. Similar to JMT, it predict tasks in different layers. In a lower layer r, it predicts the POS tags and predicate type. Then in a middle layer p, it predicts the dependency head of each word. Finally, it predicts the semantic roles with a bilinear layer. Dependency parsing has been shown an important feature for semantic role labeling. The dependency head is predicted by one attention head in self attention, thus LISA can utilize informative dependency information as well as learn other clues with the remaining heads. The model is optimized by weighted summation of losses from all tasks.

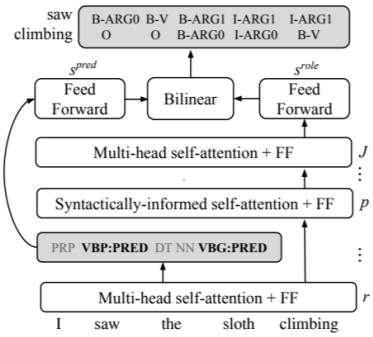


Figure 3: The architecture of Linguistically-Informed Self-Attention (LISA).

2.3 MT-DNN [13]

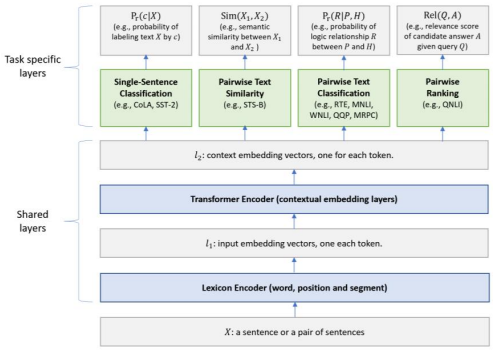


Figure 4: The architecture of Multi-Task Deep Neural Network (MT-DNN).

Multi-Task Deep Neural Network (MT-DNN) is the first paper of multi-task learning on BERT. It achieves state-of-the-art performance on GLUE benchmark [14]. Their idea is embarrassingly simple – just stacking task-specific (linear) classifiers on a shared representation. In training, instances are sampled from the union of all tasks and their losses are equally weighed. This model violates all requirements of a good multi-task learning method. Some of the tasks are weakly-related (like linguistic acceptability and sentiment analysis) which may lead to destructive interference. The loss functions are of different scales while assigned the same weights, which may cause the training process to be dominated by some losses. But it still achieves universal gain in all tasks.

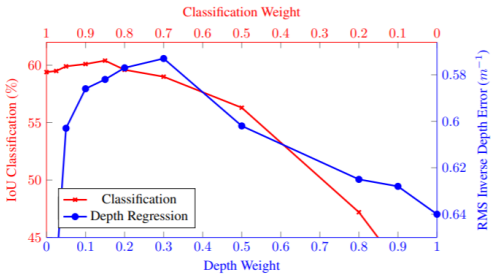
3 Learning to Multitask

Determining the sharing scheme and task weights requires lots of human efforts, which restricts the applications of multi-task learning in practice. A more efficient method is to let the model learn what to share and how to share by itself. Learning to multitask has drawn increasing attention. Existing works can be organized to *learning to weigh* and *learning to share*.

3.1 Learning to Weigh

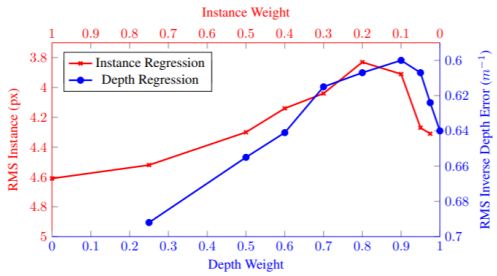
In multi-task learning, the model is jointly optimized by weighted linear sum of the losses for each individual task:

$$\min_{\{\theta_i\}_{i=1}^m} \sum_{i=1}^{m} a_i \mathcal{L}_i(\mathcal{X}_i, \mathcal{Y}_i; \theta_i) \tag{2}$$



Task Weights Class	Task Weights Depth	Class IoU [%]	Depth Err. [px]
1.0	0.0	59.4	-
0.975	0.025	59.5	0.664
0.95	0.05	59.9	0.603
0.9	0.1	60.1	0.586
0.85	0.15	60.4	0.582
0.8	0.2	59.6	0.577
0.7	0.3	59.0	0.573
0.5	0.5	56.3	0.602
0.2	0.8	47.2	0.625
0.1	0.9	42.7	0.628
0.0	1.0	-	0.640
Learned weights with task uncertainty (this work, Section 3.2)		**62.7**	**0.533**

(a) Comparing loss weightings when learning **semantic classification and depth regression**



Task Weights Instance	Task Weights Depth	Instance Err. [px]	Depth Err. [px]
1.0	0.0	4.61	
0.75	0.25	4.52	0.692
0.5	0.5	4.30	0.655
0.4	0.6	4.14	0.641
0.3	0.7	4.04	0.615
0.2	0.8	3.83	0.607
0.1	0.9	3.91	0.600
0.05	0.95	4.27	0.607
0.025	0.975	4.31	0.624
0.0	1.0		0.640
Learned weights with task uncertainty (this work, Section 3.2)		**3.54**	**0.539**

(b) Comparing loss weightings when learning **instance regression and depth regression**

Figure 5: Learning multiple tasks improves the model's representation and individual task performance.

However, there are a number of issues with this method. Namely, model performance is extremely sensitive to weight selection, as illustrated in Figure 7. These weight hyper-parameters are expensive to tune, often taking many days for each trial. Therefore, it is desirable to find a more convenient approach which is able to learn the optimal weights.

Multi Task Learning with Homoscedastic Uncertainty [15]. The homoscedastic uncertainty is an intrinsic property of tasks. It cannot be explained away with increasing training data and stays constant for all input data (for example, label noise). This work derives a weighting scheme from minimizing the empirical risk of multiple tasks. Specifically, for classification problem, we can formulate the label distribution as a Boltzmann distribution:

$$P(y|x,\sigma) = \text{softmax}(\frac{1}{\sigma^2} f(x))$$

where σ measures the uncertainty in tasks (also referred as temperature). Then for two tasks T_1 and T_2, their joint probability is:

$$\begin{aligned}
-\log P(y_1 = c_1, y_2 = c_2|x) &= -\log P(y_1 = c_1|x,\sigma_1) - \log P(y_2 = c_2|x,\sigma_2) \\
&= -\log \text{softmax}(\frac{1}{\sigma_1^2} f(x)) - \log \text{softmax}(\frac{1}{\sigma_2^2} g(x)) \\
&= \frac{1}{\sigma_1^2}\mathcal{L}_1 + \frac{1}{\sigma_2^2}\mathcal{L}_2 + \log \frac{\sum_{c'} \exp(\frac{1}{\sigma_1^2} f_{c'}(x))}{\left(\sum_{c'} \exp(f_{c'}(x))\right)^{\frac{1}{\sigma_1^2}}} + \log \frac{\sum_{c'} \exp(\frac{1}{\sigma_2^2} g_{c'}(x))}{\left(\sum_{c'} \exp(g_{c'}(x))\right)^{\frac{1}{\sigma_2^2}}} \\
&\approx \frac{1}{\sigma_1^2}\mathcal{L}_1 + \frac{1}{\sigma_2^2}\mathcal{L}_2 + \log \sigma_1 \sigma_2
\end{aligned}$$

This loss function can be extended to multi-task regression problem. The derived loss function is consistent with our intuition. When the noise σ increases, the task weight will decrease. So this method encourages the model to prefer more accurate tasks.

Dynamic Task Prioritization for Multitask Learning [16]. Instead of task uncertainty, this work weighs each task by its difficulty. Intuitively, the model should pay more attention to harder tasks and less to easier ones. Specifically, given tasks $\{T_i\}_{i=1}^m$, the model calculates a performance metric $\{\mathtt{k}\}_{i=1}^m$, where $\mathtt{k}_i \in [0,1]$ (accuracy, F1, IoU etc.). Then the tasks are weighed by focal loss [17]:

$$\min_{\{\theta_i\}_{i=1}^m} \sum_{i=1}^m F(\mathtt{k_i}, \gamma_i)\mathcal{L}_i(\mathcal{X}_i, \mathcal{Y}_i; \theta_i)$$

where

$$F(\mathtt{k_i}, \gamma_i) = -(1-\mathtt{k_i})^{\gamma_t} \log(\mathtt{k_i})$$

3.2 Learning to Share

In practice, multi-task learning involves searching an enormous space of possible parameter sharing architectures to find (a) the layers or subspaces that benefit from sharing, (b) the appropriate amount of sharing, and (c) the appropriate relative weights of the different task losses. Recently, many works focus on how to learn the sharing scheme.

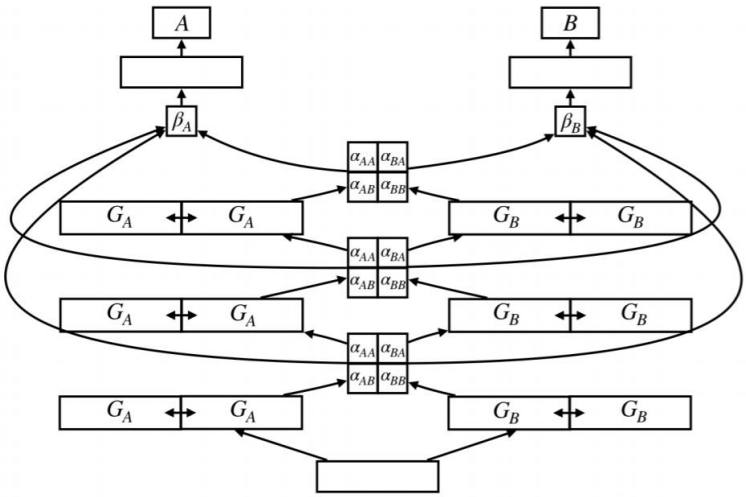


Figure 6: The architecture of sluice network.

Sluice Network [2]. The sluice network mainly solves two problems: (1) what network can be shared and what should be private, and (2) which layers should tasks be predicted in. It introduces an attention-like gate to control the information flow. Specifically, in layer k, the representations are $[h^k_{A,1}, h^k_{A,2}]$ and $[h^k_{B,1}, h^k_{B,2}]$ for task A and B respectively, then in layer $k+1$, the representations are computed by:

$$\begin{bmatrix} h^{k+1}_{A,1} \\ h^{k+1}_{A,2} \\ h^{k+1}_{B,1} \\ h^{k+1}_{B,2} \end{bmatrix} = \mathbf{W} \begin{bmatrix} h^{k}_{A,1} \\ h^{k}_{A,2} \\ h^{k}_{B,1} \\ h^{k}_{B,2} \end{bmatrix}$$

where $\mathbf{W} \in \mathcal{R}^{2\times 2}$. The final representation $\hat{h}_A$ and $\hat{h}_B$ are calculated by linear mixture of all layers:

$$\hat{h}_A = \sum_{k=1}^{L} \beta^A_k h^k_A$$

which enables the network to learn task hierarchy.

This attention-like sharing scheme is widely used in other papers, such as MTAN [13], adaptive routing block [18], and MAN-MOE [19].

3.3 Learning to Sample

In multi-task learning, we want to involve related tasks and discard weakly-related or contradicted tasks. Some works study how to sample the tasks so that relevant tasks are sampled more often.

AutoSeM [20]. The AutoSeM framework models batch sampling as a multi-armed bandit problem. Let $\{T_1, T_2, ..., T_m\}$ represent the set of tasks (or set of arms), the goal of AutoSeM is to select a sequence of tasks over the training trajectory to maximize the expected future payoff. They take the prior reward distribution of $\theta_k \in [0, 1]$ to be Beta-distributed with α_k, β_k, and the PDF of θ_k is:

$$p(\theta_k) = \frac{\Gamma(\alpha_k + \beta_k)}{\Gamma(\alpha_k)\Gamma(\beta_k)} \theta_k^{\alpha_k - 1} (1 - \theta_k)^{\beta_k - 1}$$

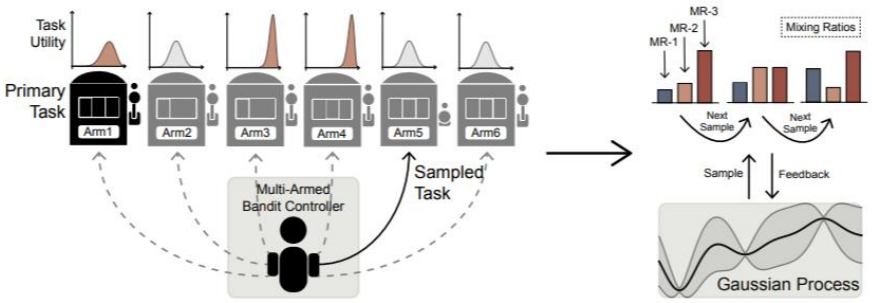


Figure 7: The AutoSeM Framework.

They formulate the reward at round t_b as a Bernoulli variable $r_{t_b} \in \{0, 1\}$, which equals 1 when the validation score improves and 0 otherwise. Then the parameters of reward distribution can be updated by:

$$(\alpha_k, \beta_k) = \begin{cases} (\alpha_k, \beta_k), & \text{if } x^s_{t_b} \neq k \\ (\alpha_k, \beta_k) + (r_{t_b}, 1 - r_{t_b}), & \text{if } x^s_{t_b} = k \end{cases}$$

where $x^s_{t_b}$ is the sampled task at round t_b.

References

[1] Yu Zhang and Qiang Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.

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

[3] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1923–1933, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[4] Jonathan Baxter. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine learning*, 28(1):7–39, 1997.

[5] Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*, 2018.

[6] Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015.

[7] Jan Niehues and Eunah Cho. Exploiting linguistic resources for neural machine translation using multi-task learning. *arXiv preprint arXiv:1708.00993*, 2017.

[8] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-yi Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 912–921, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[9] Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[10] Victor Sanh, Thomas Wolf, and Sebastian Ruder. A hierarchical multi-task approach for learning embeddings from semantic tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6949–6956, 2019.

[11] Jiang Guo, Wanxiang Che, Haifeng Wang, Ting Liu, and Jun Xu. A unified architecture for semantic role labeling and relation classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1264–1274, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

[12] Xiangyun Zhao, Haoxiang Li, Xiaohui Shen, Xiaodan Liang, and Ying Wu. A modulation module for multi-task learning with applications in image retrieval. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 401–416, 2018.

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

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

[15] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7482–7491, 2018.

[16] Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. Dynamic task prioritization for multitask learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 270–287, 2018.

[17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[18] Poorya Zaremoodi, Wray Buntine, and Gholamreza Haffari. Adaptive knowledge sharing in multi-task learning: Improving low-resource neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 656–661, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[19] Xilun Chen, Ahmed Hassan Awadallah, Hany Hassan, Wei Wang, and Claire Cardie. Multi-source cross-lingual model transfer: Learning what to share. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3098–3112, Florence, Italy, July 2019. Association for Computational Linguistics.

[20] Han Guo, Ramakanth Pasunuru, and Mohit Bansal. Autosem: Automatic task selection and mixing in multi-task learning. *arXiv preprint arXiv:1904.04153*, 2019.