Systems	POS	NER	Chunk.
Single Task Models:			
? (**?**)	97.29	89.59	**94.32**
? (**?**)	97.25	89.91	93.67
? (**?**)	**97.30**	**90.08**	93.71
Multi-Task Models:			
Hard sharing†	97.23	90.38	94.32
Meta-MTL-LSTM†	97.45	90.72	95.11
Sparse sharing (ours)	**97.57**	**90.87**	**95.26**

Table 5: Experimental results on Exp3. POS, NER and Chunking tasks come from PTB, CoNLL-2003, CoNLL-2000 respectively. † denotes the model is implemented by **?** (**?**). All listed models are equipped with CRF.

task-specific layers in Exp3.

Where does the benefit come from? To figure out where the benefit of sparse sharing comes from, we evaluate the performance of generated subnets on their corresponding tasks in the single-task setting. As shown in the second row of Table 3, each tasks' subnet does not significantly improve the performance on that task. Thus the benefit comes from shared knowledge in other tasks, rather than pruning.

Avoiding negative transfer. Our experimental results show that multi-task learning does not always yield improvements. Sometimes it even hurts the performance on some tasks, as shown in Table 3, which is called *negative transfer*. Surprisingly, we find that sparse sharing mechanism is helpful to avoid negative transfer.

Analysis and Discussions

In this section, we further analyze the ability of sparse sharing on negative transfer, task interaction and sparsity. At last, an ablation study about multi-task warmup is conducted.

About Negative Transfer

Negative effects usually occurs when there are unrelated tasks. To further analyze the impact of negative transfer for our model, we construct two loosely related tasks. One task is the real CoNLL-2003 NER task. The other task, Position Prediction (PP), is synthetic. The PP task is to predict each token's position in the sentence. The PP task annotation is collected from CoNLL-2003. Thus, the PP task and CoNLL-2003 NER are two loosely related tasks and form a multi-task setting.

We employ a 1-layer BiLSTM with 50-dimensional hidden size as shared module and a fully-connected layer as task-specific layer. Our word embeddings are in 50 dimensions and randomly initialized. Our experimental results are shown in Table 6. Note that Δ is defined as the increase of performance compared with single-task models. The synthetic experiment shows that hard sharing suffers from negative transfer, while sparse sharing does not.

	NER	Δ	PP	Δ
Single task	71.05	-	99.21	-
Hard sharing	61.62	-9.43	99.50	$+0.29$
Sparse sharing	71.46	$+0.41$	99.45	$+0.24$

Table 6: Results of the synthetic experiment. The performance deterioration due to negative transfer is in gray.

About Task Relatedness

Furthermore, we quantitatively discuss the task relatedness and its relationship with multi-task benefit. We provide a novel perspective to analyze task relatedness. We define the Overlap Ratio (OR) among mask matrices as the zero-norm of their intersection divided by the zero-norm of their union:

$$\mathrm{OR}(M_1, M_2, \cdots, M_T) = \frac{\| \cap_{t=1}^{T} M_t \|_0}{\| \cup_{t=1}^{T} M_t \|_0}. \quad (4)$$

Each mask matrix is associated with a subnet. On the one hand, OR reflects the similarity among subnets. On the other hand, OR reflects the degree of sharing among tasks.

We group the three tasks on CoNLL-2003 into pairs and evaluate the performance of sparse and hard sharing on these task pairs. As shown in Table 7, we find that the larger the mask OR, which means the more correlated the two tasks are, the smaller the improvement of sparse sharing compared with hard sharing. This suggests that when tasks are closely related, hard sharing is still an efficient parameter sharing mechanism.

Task Pairs	Mask OR	$\Delta(S^2 - HS)$
POS & NER	0.18	0.4
NER & Chunking	0.20	0.34
POS & Chunking	0.50	0.05

Table 7: Mask Overlap Ratio (OR) and the improvement for sparse sharing (S^2) compared to hard sharing (HS) of tasks on CoNLL-2003. The improvement is calculated using the average performance on the test set.

About Sparsity

In addition, we evaluate various combinations of subnets with different sparsity. For the sake of simplicity, we select subnet for each task with the same sparsity to construct the sparse sharing architecture. Figure 4 plots the average test performance and mask OR with different sparsity combinations. Our evaluation is carried out on CoNLL-2003.

When $M_1 = M_2 = M_3 = \mathbf{1}$, the sparsity of subnets for the three tasks is 100%, and the mask OR takes 1. In this case, sparse sharing is equivalent to hard sharing. With the decrease of sparsity, the mask OR also decreases while the average performance fluctuates.

About Multi-Task Warmup

At last, we conduct an ablation study about multi-task warmup (MTW). The performance achieved by the model

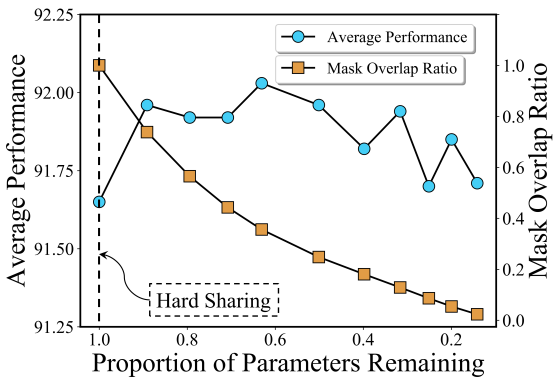

Figure 4: Average test performance and mask OR with different sparsity combinations.

with and without MTW are listed in Table 8. In most instances, the model with MTW achieves better performance.

	POS	NER	Chunking
CoNLL-2003			
Sparse sharing	95.56	90.35	91.55
− MTW	95.36	89.62	91.04
OntoNotes 5.0			
Sparse sharing	97.54	83.42	95.56
− MTW	97.53	81.15	95.48

Table 8: Test accuracy (for POS) and F1 (for NER and Chunking) on CoNLL-2003 and OntoNotes 5.0. MTW: Multi-Task Warmup.

Related Work

There are two lines of research related to our work – deep multi-task learning and sparse neural networks. Neural based multi-task learning approaches can be roughly grouped into three folds: (1) hard sharing, (2) hierarchical sharing, and (3) soft sharing. Hard sharing finds for a representation that is preferred by as many tasks as possible (**?**). In spite of its simple and parameter-efficient, it is only guaranteed to work for closely related tasks (**?**). Hierarchical sharing approaches put different level of tasks on the different network layer (**?**; **?**), which to some extent, relax the constraint about task relatedness. However, the hierarchy of tasks is usually designed by hand through the skill and insights of experts. Besides, tasks can hurt each other when they are embedded into the same hidden space (**?**). To mitigate negative transfer, soft sharing is proposed and achieves success in computer vision (**?**) and natural language processing (**?**; **?**). In spite of its flexible, soft sharing allows each task to have its separate parameters, thus is parameter inefficient. Different from these work, sparse sharing is a fine-grained parameter sharing strategy that is flexible to handle heterogeneous tasks and parameter efficient.

Our approach is also inspired by the sparsity of networks, especially the Lottery Ticket Hypothesis (**?**). **?** (**?**) finds that a subnet (winning ticket) – that can reach test accuracy comparable to the original network – can be produced through Iterative Magnitude Pruning (IMP). Further, **?** (**?**) introduce late resetting to stabilize the lottery ticket hypothesis at scale. Besides, **?** (**?**) confirms that winning ticket initializations exist in LSTM and NLP tasks. Our experiments also demonstrate that winning tickets do exist in sequence labeling tasks. In addition, it is worth noticing that sparse sharing architecture is also possible to be learned using variational dropout (**?**), l_0 regularization (**?**) or other pruning techniques.

Conclusion

Most existing neural-based multi-task learning models are done with parameter sharing, e.g. hard sharing, soft sharing, hierarchical sharing etc. These sharing mechanisms have some inevitable limitations: (1) hard sharing struggles with heterogeneous tasks, (2) soft sharing is parameter-inefficient, (3) hierarchical sharing depends on manually design. To alleviate the limitations, we propose a novel parameter sharing mechanism, named Sparse Sharing. The parameters in sparse sharing architectures are partially shared across tasks, which makes it flexible to handle loosely related tasks. To induce such architectures, we propose a simple yet efficient approach that can automatically extract subnets for each task. The obtained subnets are overlapped and trained in parallel. Our experimental results show that sparse sharing architectures achieve consistent improvement while requiring fewer parameters. Besides, our synthetic experiment shows that sparse sharing avoids negative transfer even when tasks are unrelated.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work was supported by the National Key Research and Development Program of China (No. 2018YFC0831103), National Natural Science Foundation of China (No. 61672162), Shanghai Municipal Science and Technology Major Project (No. 2018SHZDZX01) and ZJLab.

Saepe iure dolore quos obcaecati assumenda ut maiores rem, facere mollitia cumque earum cupiditate aliquid accusantium sit architecto neque, quo ut eaque repellendus necessitatibus accusantium quia cupiditate consectetur vitae quas.Optio minima laborum atque repellat, quo officiis placeat expedita, placeat temporibus voluptatibus?Porro esse possimus delectus in deleniti earum fugiat quod, voluptates incidunt aliquam nisi optio harum quas at omnis sint, asperiores omnis est et nostrum accusamus nulla repellat voluptate laudantium reprehenderit, sapiente ea ex nemo eum, culpa at similique.Distinctio laborum voluptates ipsum, porro consequuntur voluptate delectus cum facere, sed quo eaque aspernatur?Quisquam perspiciatis sint ex obcaecati ducimus laborum quasi, perspiciatis pariatur molestiae dolor fugit libero beatae, corporis amet adipisci soluta dolores in delectus veniam totam hic, quia