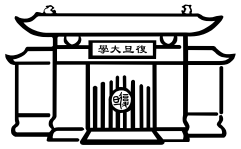
復旦大學



Learning Sparse Sharing Architectures for Multiple Tasks

Tianxiang Sun*, Yunfan Shao*, Xiaonan Li, Pengfei Liu,
Hang Yan, Xipeng Qiu, Xuanjing Huang

txsun19@fudan.edu.cn

New York City, 2020/02/09

Outline

Multi-Task Learning: Sharing Mechanisms

Sparse Sharing Mechanism

Approach: Learning Sparse Sharing Architectures

Experiments

Analysis and Discussions

Conclusion

Outline

Multi-Task Learning: Sharing Mechanisms

Sparse Sharing Mechanism

Approach: Learning Sparse Sharing Architectures

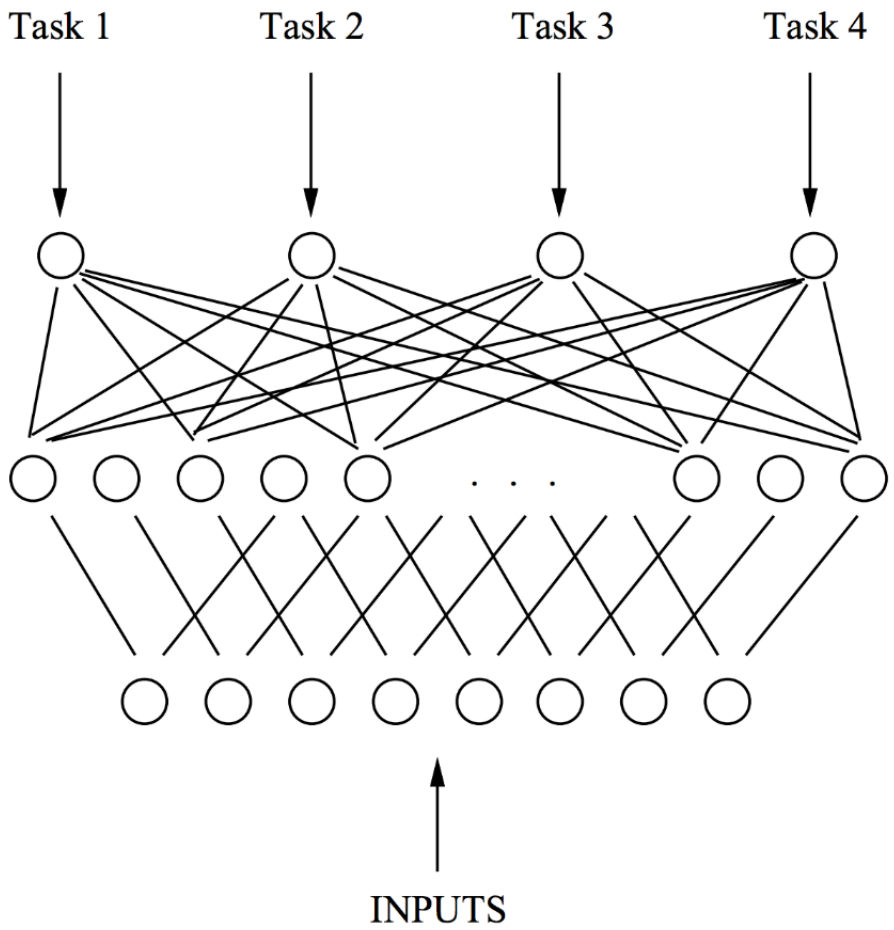
Experiments

Analysis and Discussions

Conclusion

What is Multi-Task Learning(MTL)?

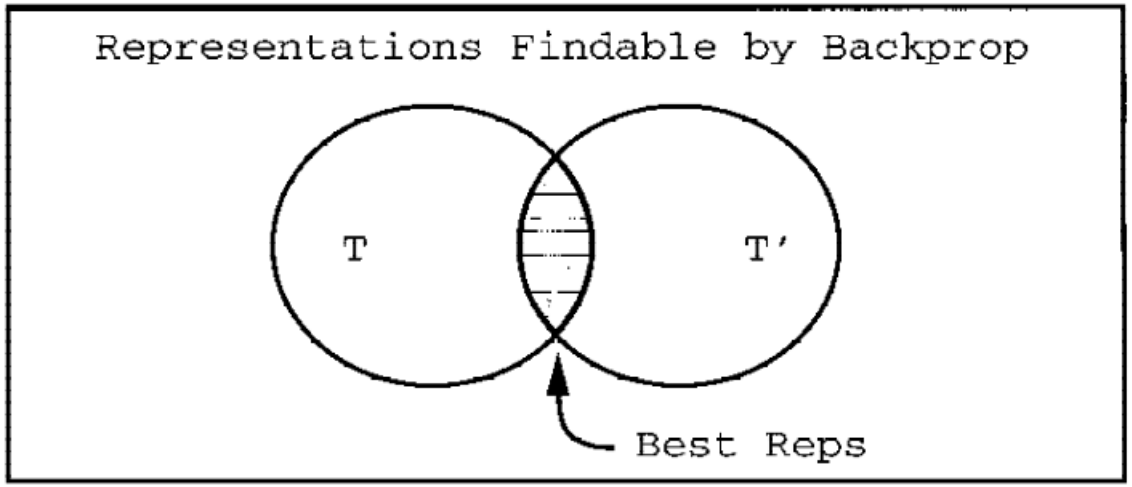
"Multi-task Learning is an approach to inductive transfer that improves generalization by using the domain information contained in the training signals of related tasks as an inductive bias."



Caruana, R. 1997. Multitask learning. *Machine Learning* 28(1):41–75.

How does MTL work?

- Representation Bias (Inductive Bias)



Caruana, R. 1997. Multitask learning. *Machine Learning* 28(1):41–75.

Formulation

- T tasks: $\mathcal{D}_t = \{x_n^t, y_n^t\}_{n=1}^{N_t}$
- Shared layers $\mathcal{E}$ parameterized by $\theta_{\mathcal{E}} = \{\theta_{\mathcal{E},1}, \ldots, \theta_{\mathcal{E},L}\}$
- Task-specific layers $\mathcal{F}^t$ parameterized by $\theta_{\mathcal{F}}^t$
- Parameters: $\theta = (\theta_{\mathcal{E}}, \theta_{\mathcal{F}}^1, \ldots, \theta_{\mathcal{F}}^T)$
- Objective: $\mathcal{L}(\theta) = \sum_{t=1}^{T} \lambda_t \sum_{n=1}^{N_t} \mathcal{L}_t(\hat{y}_n^t, y_n^t)$

Multi-Task Sharing Mechanisms

MTL is typically done with *parameter sharing*:

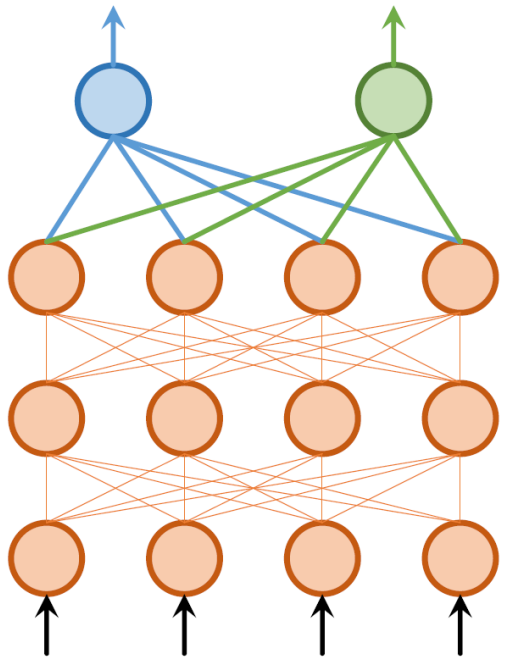
- Hard Sharing (Collobert and Weston 2008; Subramanian et al. 2018; Liu et al. 2019)
- Soft Sharing (Misra et al. 2016; Ruder et al. 2019)
- Hierarchical Sharing (Søgaard and Goldberg 2016; Hashimoto et al. 2017)
- ...

Hard Sharing

- Stack the task-specific layers on the top of the shared layers
- Inference: $\hat{y}_n^t = \mathcal{F}^t(\mathcal{E}(x_n^t; \theta_\mathcal{E}); \theta_\mathcal{F}^t)$
- Advantages:
  1. easy to implement
  2. parameter efficient
- Disadvantages:

  Struggle with loosely related/unrelated tasks

  (Negative Transfer)

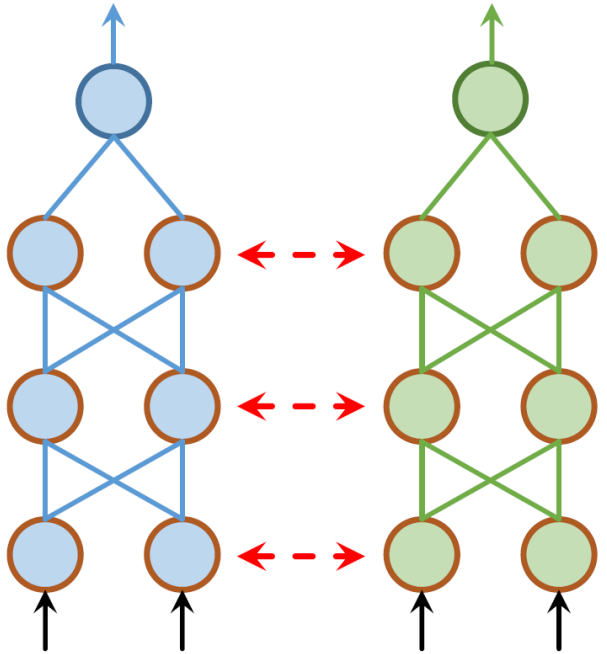


Soft Sharing

- Each task has separate model and parameters, but each model can access the information inside other models
- Advantages:

  Makes no assumptions about task relatedness
- Disadvantages:

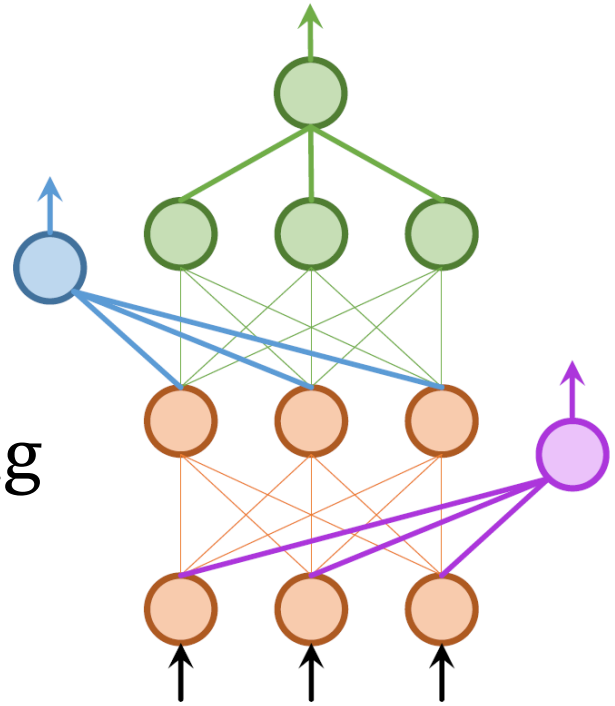
  Not parameter-efficient



Hierarchical Sharing

- Put different task supervisions at different layers
- Inference: $\hat{y}_n^t = \mathcal{F}^t(\mathcal{E}(x_n^t; \theta_{\mathcal{E}(1:l)}); \theta_{\mathcal{F}}^t)$
- Advantages:
  1. more flexible than hard sharing
  2. more parameter-efficient than soft sharing
- Disadvantages:

  Hard to design an effective hierarchy



Limitations of Existing Sharing Mechanisms

- **Hard sharing**: Struggle with loosely related tasks
- **Hierarchical sharing**: Dependent on manual design
- **Soft sharing**: Parameter-inefficient

Motivation

Does there exist a multi-task sharing mechanism:

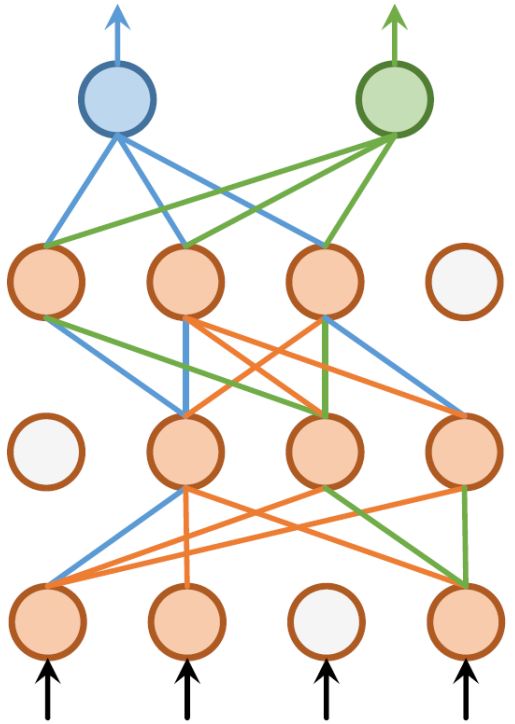
1. It is compatible with a wide range of tasks, regardless of whether the tasks are related or not.
2. It does not depend on manually designing the sharing structure based on characteristic of tasks.
3. It is parameter efficient.

Outline

Sparse Sharing Mechanism

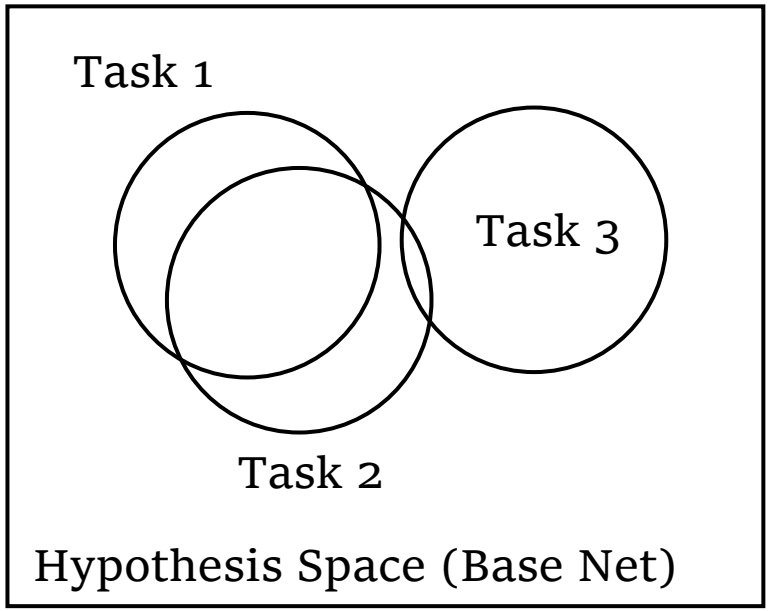
- Base Network: $\mathcal{E}$
- Assign each task a subnet
- Subnet: $\mathcal{E}^t(x) = \mathcal{E}(x; M_t \odot \theta_{\mathcal{E}})$
- Hard sharing $\rightarrow M_t = \mathbf{1}$
- Hierarchical sharing $\rightarrow$

$$\theta_{\mathcal{E}} = \{\theta_{\mathcal{E},1}, \theta_{\mathcal{E},2}\} \quad M_1 = \{\mathbf{1}, \mathbf{0}\} \quad M_2 = \{\mathbf{1}, \mathbf{1}\}$$



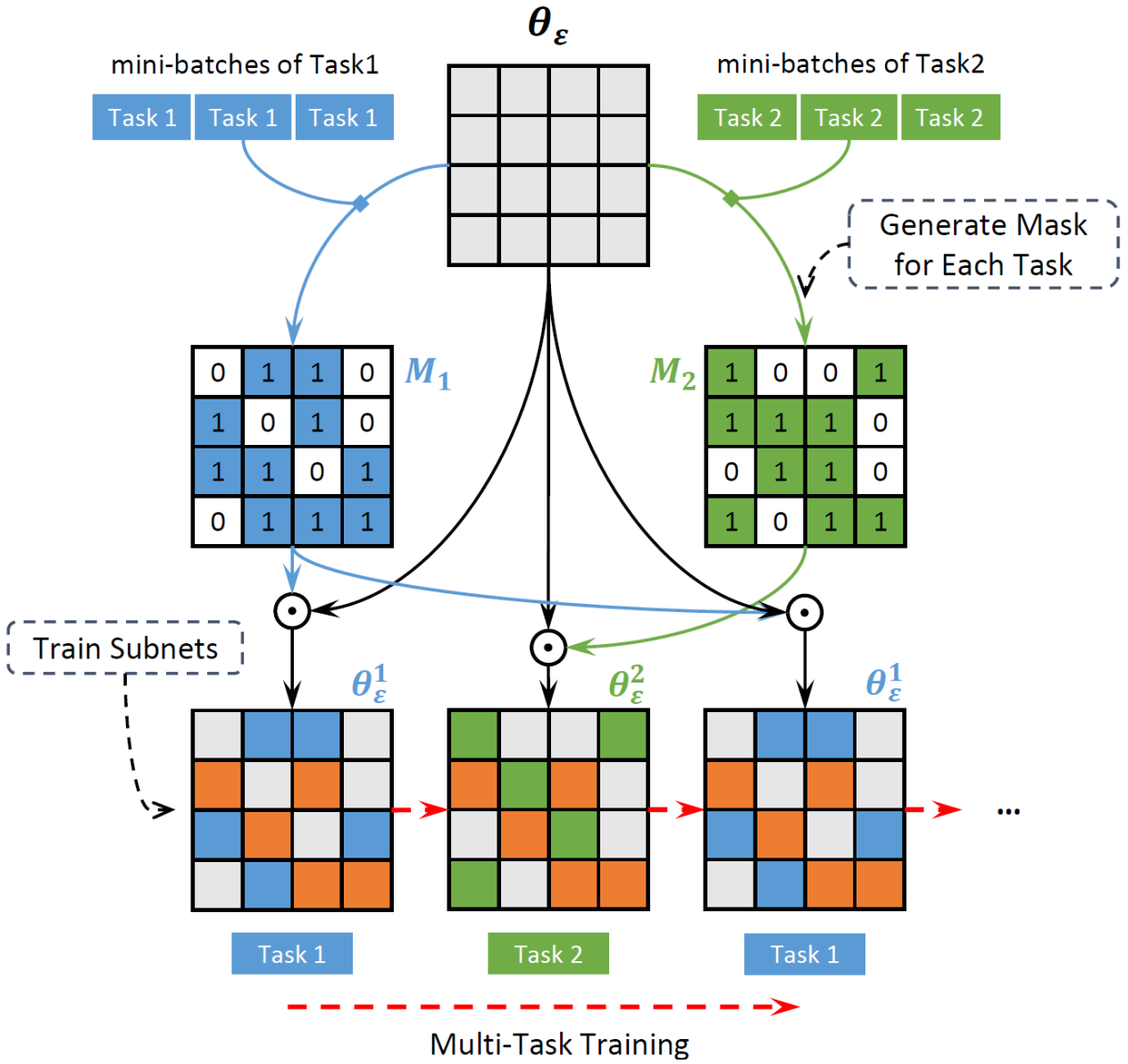
Views of Sparse Sharing

- Over-parameterized base net → Large hypothesis space
- Subnet → Hypothesis subspace
- Inductive bias → Subnet structure
- Parameter overlap → Task relatedness
- Biologically intuitive:
  1. Sparse topology (Pessoa 2014)
  2. Different subnets for different tasks (MacLeod 1991)



Outline

Overview of Our Approach



Generating Subnets for Each Task

- ## Iterative Magnitude Pruning (IMP)

  proposed in (Frankle and Carbin 2019) (ICLR'2019 best paper)

1. Randomly initialize a neural network $f(x;\theta_0)$ (where $\theta_0 \sim \mathcal{D}_\theta$).
2. Train the network for j iterations, arriving at parameters θ_j.
3. Prune $p\%$ of the parameters in θ_j, creating a mask m.
4. Reset the remaining parameters to their values in θ_0, creating the winning ticket $f(x; m \odot \theta_0)$.

Generating Subnets for Each Task

➢ Iterative Magnitude Pruning (IMP)

Algorithm 1 Sparse Sharing Architecture Learning

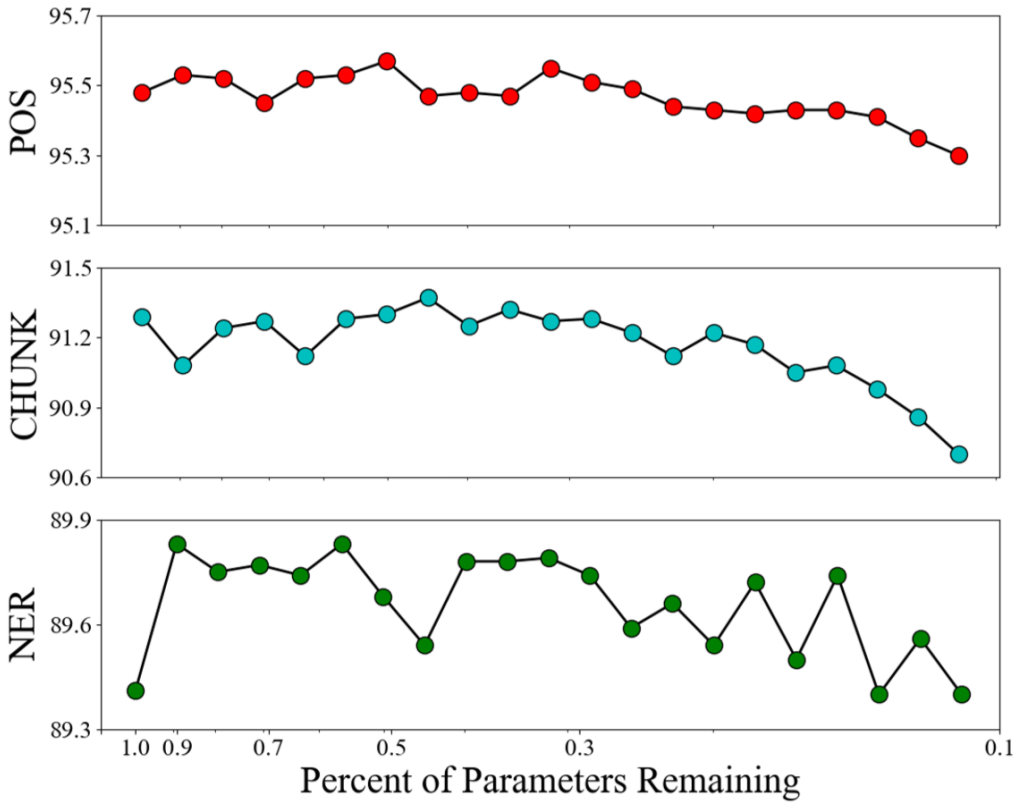
Require: Base Network $\mathcal{E}$; Pruning rate α; Minimal sparsity S; Datasets for T tasks $\mathcal{D}_1, \cdots, \mathcal{D}_T$, where $\mathcal{D}_t = \{x_n^t, y_n^t\}_{n=1}^{N_t}$.

1: Randomly initialize $\theta_{\mathcal{E}}$ to $\theta_{\mathcal{E}}^{(0)}$.
2: **for** $t = 1 \cdots T$ **do**
3: Initialize mask $M_t^z = \mathbf{1}^{|\theta_{\mathcal{E}}|}$, where $z = 1$.
4: Train $\mathcal{E}(x; M_t^z \odot \theta_{\mathcal{E}})$ for k steps with data sampled from $\mathcal{D}_t$, producing network $\mathcal{E}(x; M_t^z \odot \theta_{\mathcal{E}}^{(k)})$. Let $z \leftarrow z + 1$.
5: Prune α percent of the remaining parameters with the lowest magnitudes from $\theta_{\mathcal{E}}^{(k)}$. That is, let $M_t^z[j] = 0$ if $\theta_{\mathcal{E}}^{(k)}[j]$ is pruned.
6: If $\frac{\|M_t^z\|_0}{|\theta_{\mathcal{E}}|} \leq S$, the masks for task t are $\{M_t^i\}_{i=1}^z$.
7: Otherwise, reset $\theta_{\mathcal{E}}$ to $\theta_{\mathcal{E}}^{(0)}$ and repeat steps 4-6 iteratively to learn more sparse subnetwork.
8: **end for**
9: **return** $\{M_1^i\}_{i=1}^z, \{M_2^i\}_{i=1}^z, \cdots \{M_T^i\}_{i=1}^z$.

Select Subnets

- Pick the subnet that performs best on the dev set.
- If there are multiple best-performing subnets, take the subnet with the lowest sparsity.

POS	CHUNK	NER
50.12%	44.67%	56.23%



Training Subnets in Parallel

$$\mathcal{L}(\theta) = \sum_{t=1}^{T} \lambda_t \sum_{n=1}^{N_t} \mathcal{L}_t(\hat{y}_n^t, y_n^t)$$

1. Select the next task t.
   - Proportional sampling (Sanh, Wolf, and Ruder 2019)
2. Select a random mini-batch for task t.
3. Feed this batch of data into the subnetwork corresponding to task t, i.e. $\mathcal{E}(x; M_t \odot \theta_{\mathcal{E}})$.
4. Update the subnetwork parameters for this task by taking a gradient step with respect to this mini-batch.
5. Go to 1.

Multi-Task Warmup (MTW)

Algorithm 1 Sparse Sharing Architecture Learning

Require: Base Network $\mathcal{E}$; Pruning rate α; Minimal sparsity S; Datasets for T tasks $\mathcal{D}_1, \cdots, \mathcal{D}_T$, where $\mathcal{D}_t = \{x_n^t, y_n^t\}_{n=1}^{N_t}$.

1: Randomly initialize $\theta_{\mathcal{E}}$ to $\theta_{\mathcal{E}}^{(0)}$. MTW: $\theta_{\mathcal{E}}^{(0)} \rightarrow \theta_{\mathcal{E}}^{(w)}$

2: **for** $t = 1 \cdots T$ **do**

3: Initialize mask $M_t^z = \mathbf{1}^{|\theta_{\mathcal{E}}|}$, where $z = 1$.

4: Train $\mathcal{E}(x; M_t^z \odot \theta_{\mathcal{E}})$ for k steps with data sampled from $\mathcal{D}_t$, producing network $\mathcal{E}(x; M_t^z \odot \theta_{\mathcal{E}}^{(k)})$. Let $z \leftarrow z + 1$.

5: Prune α percent of the remaining parameters with the lowest magnitudes from $\theta_{\mathcal{E}}^{(k)}$. That is, let $M_t^z[j] = 0$ if $\theta_{\mathcal{E}}^{(k)}[j]$ is pruned.

6: If $\frac{\|M_t^z\|_0}{|\theta_{\mathcal{E}}|} \leq S$, the masks for task t are $\{M_t^i\}_{i=1}^z$.

7: Otherwise, reset $\theta_{\mathcal{E}}$ to $\theta_{\mathcal{E}}^{(w)}$ and repeat steps 4-6 iteratively to learn more sparse subnetwork.

8: **end for**

9: **return** $\{M_1^i\}_{i=1}^z, \{M_2^i\}_{i=1}^z, \cdots \{M_T^i\}_{i=1}^z$.

Outline

Experiments

- Tasks: Part-of-Speech, NER, Chunking
- Datasets

  Exp1: CoNLL-2003

  Exp2: OntoNotes 5.0

  Exp3: PTB + CoNLL-2003 + CoNLL-2000
- Model Settings

  Base model: CNN-BiLSTM (Ma and Hovy 2016)

  Multi-Task baselines: hard/soft/hierarchical sharing

Exp1 & Exp2

Systems	POS		NER		Chunking		# Params
	Test Acc.	Δ	Test F1	Δ	Test F1	Δ	
Exp1: CoNLL-2003							
Single task	95.09	-	89.36	-	89.92	-	1602k
Single task (subnet)	95.11	+0.02	89.39	+0.03	89.96	+0.04	811k
Hard sharing	95.34	+0.25	88.68	−0.68	90.92	+1.00	534k
Soft sharing	95.16	+0.07	89.35	−0.01	90.71	+0.79	1596k
Hierarchical sharing	95.09	+0.00	89.30	−0.06	90.89	+0.97	1497k
Sparse sharing (ours)	**95.56**	+0.47	**90.35**	+0.99	**91.55**	+1.63	**396k**
Exp2: OntoNotes 5.0							
Single task	97.40	-	82.72	-	95.21	-	4491k
Single task (subnet)	97.42	+0.02	82.94	+0.22	95.28	+0.07	1459k
Hard sharing	97.46	+0.06	82.95	+0.23	95.52	+0.31	1497k
Soft sharing	97.34	−0.06	81.93	−0.79	95.29	+0.08	4485k
Hierarchical sharing	97.22	−0.18	82.81	+0.09	95.53	+0.32	1497k
Sparse sharing (ours)	**97.54**	+0.14	**83.42**	+0.70	**95.56**	+0.35	**662k**

Exp1 & Exp2

Systems	POS Test Acc.	Δ	NER Test F1	Δ	Chunking Test F1	Δ	# Params
Exp1: CoNLL-2003							
Single task	95.09	-	89.36	-	89.92	-	1602k
Single task (subnet)	95.11	+0.02	89.39	+0.03	89.96	+0.04	811k
Hard sharing	95.34	+0.25	88.68	−0.68	90.92	+1.00	534k
Soft sharing	95.16	+0.07	89.35	−0.01	90.71	+0.79	1596k
Hierarchical sharing	95.09	+0.00	89.30	−0.06	90.89	+0.97	1497k
Sparse sharing (ours)	**95.56**	+0.47	**90.35**	+0.99	**91.55**	+1.63	**396k**
Exp2: OntoNotes 5.0							
Single task	97.40	-	82.72	-	95.21	-	4491k
Single task (subnet)	97.42	+0.02	82.94	+0.22	95.28	+0.07	1459k
Hard sharing	97.46	+0.06	82.95	+0.23	95.52	+0.31	1497k
Soft sharing	97.34	−0.06	81.93	−0.79	95.29	+0.08	4485k
Hierarchical sharing	97.22	−0.18	82.81	+0.09	95.53	+0.32	1497k
Sparse sharing (ours)	**97.54**	+0.14	**83.42**	+0.70	**95.56**	+0.35	**662k**

Outline

About Negative Transfer

In the future studies, there are several issues to be addressed. Firstly, outlier tasks, which are unrelated to other tasks, are well known to hamper the performance of all the tasks when learning them jointly. There are some methods to alleviate negative effects outlier tasks bring. However, there lacks principled ways and theoretical analyses to study the resulting negative effects. In order to make MTL safe to be used by human, this is an important issue and needs more studies.

Zhang, Y., & Yang, Q. 2017. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*.

About Negative Transfer

- Construct an unrelated multi-task setting
  - Real: Named Entity Recognition (NER)
  - Synthetic: Position Prediction (PP)

	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

About Task Relatedness

- Define mask overlap ratio (OR) as:

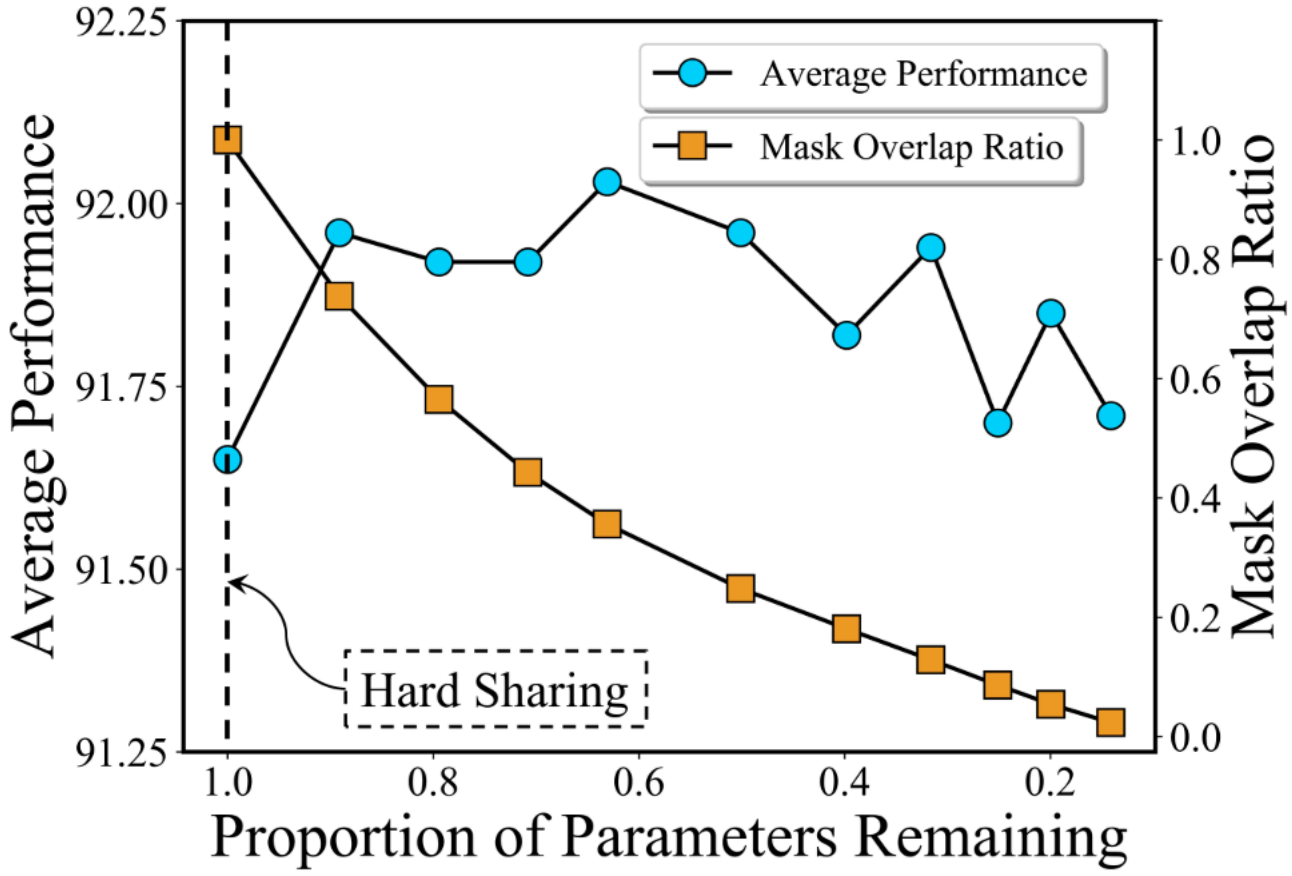
$$\mathrm{OR}(M_1, M_2, \cdots, M_T) = \frac{\| \cap_{t=1}^{T} M_t \|_0}{\| \cup_{t=1}^{T} M_t \|_0}$$

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

- Combinations of subnets with different sparsity



Outline

Conclusion

- Does sparse sharing architecture meets the requirements?
  1. It is compatible with a wide range of tasks, regardless of whether the tasks are related or not.
  2. It does not depend on manually designing the sharing structure based on characteristic of tasks.
  3. It is parameter efficient.
- It seems YES!

Thanks !

Q & A

txsun19@fudan.edu.cn

Sequence Labeling Tasks

- POS, NER and Chunking

Words	Results	of	South	Korean
POS	NNS	IN	JJ	JJ
NER	O	O	B-MISC	I-MISC
Chunk.	B-NP	B-PP	B-NP	I-NP

CNN-BiLSTM: A popular architecture in sequence labeling tasks