Learning what to share between loosely related tasks

Sebastian Ruder[12]*, **Joachim Bingel**[3], **Isabelle Augenstein**[4]*, **Anders Søgaard**[3]
[1]Insight Research Centre, National University of Ireland, Galway
[2]Aylien Ltd., Dublin, Ireland
[3]Department of Computer Science, University of Copenhagen, Denmark
[4]Department of Computer Science, UCL, UK
sebastian@ruder.io, {bingel|soegaard}@di.ku.dk, augenstein@di.ku.dk

Abstract

Multi-task learning is motivated by the observation that humans bring to bear what they know about related problems when solving new ones. Similarly, deep neural networks can profit from related tasks by sharing parameters with other networks. However, humans do not consciously decide to transfer knowledge between tasks. In Natural Language Processing (NLP), it is hard to predict if sharing will lead to improvements, particularly if tasks are only loosely related. To overcome this, we introduce SLUICE NETWORKS, a general framework for multi-task learning where trainable parameters control the amount of sharing. Our framework generalizes previous proposals in enabling sharing of all combinations of subspaces, layers, and skip connections. We perform experiments on three task pairs, and across seven different domains, using data from OntoNotes 5.0, and achieve up to 15% average error reductions over common approaches to multi-task learning. We show that a) label entropy is predictive of gains in sluice networks, confirming findings for hard parameter sharing and b) while sluice networks easily fit noise, they are robust across domains in practice.

1 Introduction

Existing theory mainly provides guarantees for multi-task learning (MTL) of homogeneous tasks, such as pure regression or classification tasks (Baxter, 2000; Ben-David and Schuller, 2003). These guarantees, however, do not hold for the heterogeneous tasks to which multi-task learning is most often applied to in NLP, which only share a common set of input variables (Collobert and Weston, 2008; Søgaard and Goldberg, 2016). To compensate for the lack of theory in the case of these *loosely related* tasks, researchers have started to explore multi-task learning from a more experimental point of view, correlating performance gains with task properties to achieve a better understanding of when models can profit from auxiliary tasks (Bingel and Søgaard, 2017; Martínez Alonso and Plank, 2017). While such works have shed partial light on the effectiveness of particular approaches to multi-task learning, it remains hard to predict what parts of networks benefit from sharing, and to what extent they do so.

Our limited understanding of multi-task learning is also a practical problem: With hundreds of potential sharing structures, an exhaustive exploration of the search space of multi-task learning for specific problems is infeasible. Existing work (Kumar and Daumé III, 2012; Maurer et al., 2013) uses sparsity to share task predictors, but has only been applied to homogeneous tasks. Previous work in multi-task learning of heterogeneous tasks only considers at most a couple of architectures for sharing (Søgaard and Goldberg, 2016; Peng and Dredze, 2016; Martínez Alonso and Plank, 2017). In contrast, we present a framework that unifies such different approaches by introducing trainable parameters for the components that differentiate multi-task learning approaches. We build on recent work trying to learn where to split merged networks (Misra et al., 2016), as well as work trying to learn how best to combine private and shared subspaces (Bousmalis et al., 2016; Liu et al., 2017).

*Work done during a visit at the University of Copenhagen.

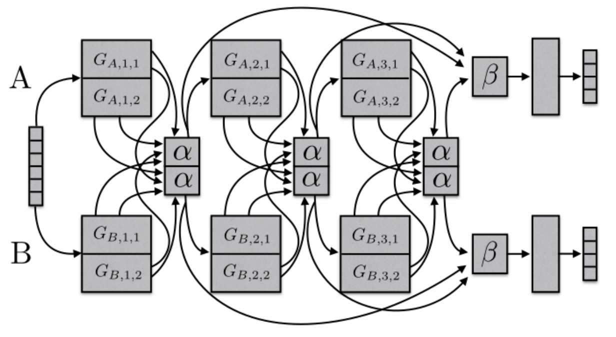

Figure 1: A SLUICE NETWORK with one main task A and one auxiliary task B. It consists of a shared input layer (shown left), two task-specific output layers (right), and three hidden layers per task, each partitioned into two subspaces. α parameters control which subspaces are shared between main and auxiliary task, while β parameters control which layer outputs are used for prediction.

Contributions Our architecture (shown in Figure 1) is empirically justified and deals with the *dirtiness* (Jalali et al., 2010) of loosely related tasks. It goes significantly beyond previous work, learning both what layers to share, and which parts of those layers to share, as well as using skip connections to learn a mixture model at the architecture's outer layer. We show that it is a generalization of various multi-task learning algorithms such as hard parameter sharing (Caruana, 1998), low supervision (Søgaard and Goldberg, 2016), and cross-stitch networks (Misra et al., 2016), as well as transfer learning algorithms such as frustratingly easy domain adaptation (Daumé III, 2007). Moreover, we study what task properties predict gains, and what properties correlate with learning certain types of sharing, as well as the inductive bias of the resulting architecture.

2 An Architecture for Learning to Share

We introduce a novel architecture for multi-task learning, which we refer to as a SLUICE NETWORK, sketched in Figure 1 for the case of two tasks. The network learns to share parameters between deep recurrent neural networks (RNNs) (Hochreiter and Schmidhuber, 1997). The recurrent networks could easily be replaced with multi-layered perceptrons or convolutional neural networks for other applications.

The two networks A and B share an embedding layer associating the elements of an input sequence, in our case English words, with vector representations via word and character embeddings. The two sequences of vectors are then passed on to their respective inner recurrent layers. Each layer is divided into subspaces, e.g., for A into $G_{\text{A},1,1}$ and $G_{\text{A},1,2}$, which allow the network to learn task-specific and shared representations, if beneficial. The output of the inner layer of network A is then passed to its second layer, as well as to the second layer of network B. This traffic of information is mediated by a set of parameters α in a way such that the second layer of each network receives a weighted combination of the output of the two inner layers. The subspaces have different weights. Importantly, these weights are trainable and allow the model to learn whether to share, whether to restrict sharing to a shared subspace, etc. Finally, a weighted combination of the outputs of the outer recurrent layers $G_{\cdot,3,\cdot}$ as well as the weighted outputs of the inner layers are mediated through β parameters, which reflect a mixture over the representations at various depths of the network. In sum, sluice networks have the capacity to learn what layers and subspaces should be shared, as well as at what layers the network has learned the best representations of the input sequences.

Matrix Regularization We cast learning what to share as a *matrix regularization* problem, following (Jacob et al., 2009; Yang and Hospedales, 2017). Assume M different tasks that are loosely related, with M potentially non-overlapping datasets $\mathcal{D}_1, \dots, \mathcal{D}_M$. Each task is associated with a deep neural network with K layers $L_1, \dots L_K$. We assume that all the deep networks have the same hyper-parameters at the outset. With loosely related tasks, one task may be better modeled with one hidden layer; another one with two (Søgaard and Goldberg, 2016); some may share many parameters, while others mostly rely on task-specific representations. Our architecture is flexible enough to learn this by allocating appropriate subspaces and mediating weights starting from the *union* of the *a priori* task networks.

Let $W \in \mathbb{R}^{M \times D}$ be a matrix in which each row i corresponds to a model θ_i with D parameters. The loss that sluice networks minimize, with a penalty term Ω, is then as follows:

$$\lambda_1 \mathcal{L}_1(\mathbf{f}(x;\theta_1), y_1) + \ldots + \lambda_M \mathcal{L}_M(\mathbf{f}(x;\theta_M), y_M) + \Omega \tag{1}$$

The loss functions $\mathcal{L}_i$ are cross-entropy functions of the form $-\sum_y p(y)\log q(y)$ where y_i are the labels of task i. Note that sluice networks are not restricted to tasks with the same loss functions, but could also be applied to jointly learn regression and classification tasks. The weights λ_i determine the importance of the different tasks during training. In all our experiments, we use the same weight for all tasks.

We *explicitly* add inductive bias to the model via the regularizer Ω, which we describe in Equation 4. However, our model also *implicitly* learns regularization through multi-task learning (Caruana, 1993) mediated by the α weights, while the β weights are used to learn the parameters of the mixture functions $\mathbf{f}(\cdot)$, as detailed in the following.

Learning Matrix Regularizers We now explain how updating α parameters can lead to different matrix regularizers. Each matrix W consists of M rows where M is the number of tasks. Each row is of length D with D the number of parameters. Subvectors $L_{m,k}$ correspond to the parameters of network m at layer k. Each layer consists of two subspaces with parameters $G_{m,k,1}$ and $G_{m,k,2}$.

Recall that our architecture is partly motivated by the observation that for loosely related tasks, only certain features in specific layers should be shared, while many of the layers and subspaces may remain more task-specific (Søgaard and Goldberg, 2016). We want to learn what to share while inducing models for the different tasks. For simplicity, we ignore subspaces at first and assume only two tasks A and B. The outputs $h_{A,k,t}$ and $h_{B,k,t}$ of the k-th layer for time step t for task A and B respectively interact through what Misra et al. (2016) refer to as *cross-stitch units* α (see Figure 1). Omitting t for simplicity, the output of the α layers is:

$$\begin{bmatrix} \widetilde{h}_{\mathrm{A},k} \\ \widetilde{h}_{\mathrm{B},k} \end{bmatrix} = \begin{bmatrix} \alpha_{\mathrm{AA}} & \alpha_{\mathrm{AB}} \\ \alpha_{\mathrm{BA}} & \alpha_{\mathrm{BB}} \end{bmatrix} \begin{bmatrix} h_{\mathrm{A},k}{}^\top , & h_{\mathrm{B},k}{}^\top \end{bmatrix} \tag{2}$$

where $\widetilde{h}_{\mathrm{A},k}$ is a linear combination of the outputs that is fed to the k+1-th layer of task A, and $\begin{bmatrix} a^\top, b^\top \end{bmatrix}$ designates the stacking of two vectors $a, b \in \mathbb{R}^D$ to a matrix $M \in \mathbb{R}^{2\times D}$.

Extending the α-layers to include subspaces, for 2 tasks and 2 subspaces, we obtain an α matrix $\in \mathbb{R}^{4\times 4}$ that not only controls the interaction between the layers of both tasks, but also between their subspaces:

$$\begin{bmatrix} \widetilde{h}_{\mathrm{A}_1,k} \\ \vdots \\ \widetilde{h}_{\mathrm{B}_2,k} \end{bmatrix} = \begin{bmatrix} \alpha_{\mathrm{A}_1\mathrm{A}_1} & \cdots & \alpha_{\mathrm{B}_2\mathrm{A}_1} \\ \vdots & \ddots & \vdots \\ \alpha_{\mathrm{A}_1\mathrm{B}_2} & \cdots & \alpha_{\mathrm{B}_2\mathrm{B}_2} \end{bmatrix} \begin{bmatrix} h_{\mathrm{A}_1,k}{}^\top , & \ldots, & h_{\mathrm{B}_2,k}{}^\top \end{bmatrix} \tag{3}$$

where $h_{\mathrm{A}_1,k}$ is the output of the first subspace of the k-th layer of task A and $\widetilde{h}_{\mathrm{A}_1,k}$ is the linear combination for the first subspace of task A. The input to the $k+1$-th layer of task A is then the concatenation of both subspace outputs: $h_{\mathrm{A},k} = \begin{bmatrix} \widetilde{h}_{\mathrm{A}_1,k}, & \widetilde{h}_{\mathrm{A}_2,k} \end{bmatrix}$.

Different α weights correspond to different matrix regularizers Ω, including several ones that have been proposed previously for multi-task learning. We review those in Section 3. For now just observe that if all α-values are set to 0.25 (or any other constant), we obtain hard parameter sharing (Caruana, 1993), which is equivalent to a heavy L_0-regularizer.

Adding Inductive Bias Naturally, we can also add inductive bias to sluice networks by partially constraining the regularizer or adding to the learned penalty. Inspired by work on shared-space component analysis (Salzmann et al., 2010), we add a penalty to enforce a division of labor and discourage redundancy between shared and task-specific subspaces. While the networks can theoretically learn such a separation, an explicit constraint empirically leads to better results and enables the sluice networks to take better advantage of subspace-specific α-values. This is modeled by an orthogonality constraint (Bousmalis et al., 2016) between the layer-wise subspaces of each model:

$$\Omega = \sum_{m=1}^{M} \sum_{k=1}^{K} \| G_{m,k,1}{}^\top G_{m,k,2} \|_F^2 \tag{4}$$

where M is the number of tasks, K is the number of layers, $\|\cdot\|_F^2$ is the squared Frobenius norm, and $G_{m,k,1}$ and $G_{k,2,m}$ are the first and second subspace respectively in the k-th layer of m-th task model.

Learning Mixtures Many tasks have an implicit hierarchy that informs their interaction. Rather than predefining it (Søgaard and Goldberg, 2016; Hashimoto et al., 2017), we enable our model to learn hierarchical relations by associating different tasks with different layers if this is beneficial for learning. Inspired by advances in residual learning (He et al., 2016), we employ skip-connections from each layer, controlled using β parameters. This layer acts as a mixture model, returning a mixture of expert predictions:

$$\widetilde{h}_{\mathrm{A}}^{\top} = \begin{bmatrix} \beta_{\mathrm{A},1} \\ \cdots \\ \beta_{\mathrm{A},k} \end{bmatrix}^{\top} \begin{bmatrix} h_{\mathrm{A},1}{}^{\top}, & \ldots & h_{\mathrm{A},k}{}^{\top} \end{bmatrix} \quad (5)$$

where $h_{\mathrm{A},k}$ is the output of layer k of model A, while $\widetilde{h}_{\mathrm{A},t}$ is the linear combination of all layer outputs of model A that is fed into the final softmax layer.

Complexity Our model only adds a minimal number of additional parameters compared to single-task models of the same architecture. In our experiments, we add α parameters between all task networks. As such, they scale linearly with the number of layers and quadratically with the number of tasks and subspaces, while β parameters scale linearly with the number of tasks and the number of layers. For a sluice network with M tasks, K layers per task, and 2 subspaces per layer, we thus obtain $4KM^2$ additional α parameters and KM β parameters. Training sluice networks is not much slower than training hard parameter sharing networks, with only an 5–7% increase in training time.

3 Prior Work as Instances of Sluice Networks

The architecture is very flexible and can be seen as a generalization over several existing algorithms for transfer and multi-task learning, including (Caruana, 1998; Daumé III, 2007; Søgaard and Goldberg, 2016; Misra et al., 2016). We show how to derive each of these below.

Hard Parameter Sharing in the two networks appears if all α values are set to the same constant (Caruana, 1998; Collobert and Weston, 2008). This is equivalent to a mean-constrained ℓ_0-regularizer $\Omega(\cdot) = |\cdot|_0^{\bar{w}_i}$ and $\sum_i \lambda_i \mathcal{L}_i < 1$. If the sum of weighted losses are smaller than 1, the loss with penalty is always the highest when all parameters are shared.

Group Lasso The ℓ_1/ℓ_2 group lasso regularizer is $\sum_{g=1}^{G} ||G_{1,i,g}||_2$, a weighted sum over the ℓ_2 norms of the groups, often used to enforce subspace sharing (Zhou et al., 2010; Świrszcz and Lozano, 2012). Our architecture learns a ℓ_1/ℓ_2 group lasso over the two subspaces (with the same degrees of freedom), when all $\alpha_{A,B}$ and $\alpha_{B,A}$-values are set to 0. When the outer layer α-values are not shared, we get block communication between the networks.

Frustratingly Easy Domain Adaptation The approach to domain adaptation in (Daumé III, 2007), which relies on a shared and a private space for each task or domain, can be encoded in sluice networks by setting all $\alpha_{A,B}$- and $\alpha_{B,A}$-weights associated with $G_{i,k,1}$ to 0, while setting all $\alpha_{A,B}$-weights associated with $G_{i,k,2}$ to $\alpha_{B,B}$, and $\alpha_{B,A}$-weights associated with $G_{i,k,2}$ to $\alpha_{A,A}$. Note that Daumé III (2007) discusses three subspaces. We obtain this space if we only share one half of the second subspaces across the two networks.

Low Supervision Søgaard and Goldberg (2016) propose a model where only the inner layers of two deep recurrent works are shared. This is obtained using heavy mean-constrained L_0 regularization over the first layer $L_{i,1}$, e.g., $\Omega(W) = \sum_i^K ||L_{i,1}||_0$ with $\sum_i \lambda_i \mathcal{L}(i) < 1$, while for the auxiliary task, only the first layer β parameter is set to 1.

Cross-Stitch Networks Misra et al. (2016) introduce cross-stitch networks that have α values control the flow between layers of two convolutional neural networks. Their model corresponds to setting the α-values associated with $G_{i,j,1}$ be identical to those for $G_{i,j,2}$, and by letting all but the β-value associated with the outer layer be 0.

In our experiments, we include hard parameter sharing, low supervision, and cross-stitch networks as baselines. We do not report results for group lasso and frustratingly easy domain adaptation, which were consistently inferior on development data by some margin.

	Domains													
	Broadcast conversation (bc)		Broadcast news (bn)		Magazines (mz)		Newswire (nw)		Pivot corpus (pc)		Telephone conversation (tc)		Weblogs (wb)	
	# sent	# words	# sent	# words	# sent	# words	# sent	# words	# sent	# words	# sent	# words	# sent	# words
Train	11846	173289	10658	206902	6905	164217	34944	878223	21520	297049	11274	90403	16734	388851
Dev	2112	29957	1292	25271	641	15421	5893	147955	1780	25206	1367	11200	2297	49393
Test	2206	35947	1357	26424	779	17874	2326	60756	1869	25883	1306	10916	2281	52225

Table 1: Number of sentences and words for the splits of each domain in the OntoNotes 5.0 dataset.

WORDS	Abramov	had	a	car	accident
CHUNK	O	B-VP	B-NP	I-NP	I-NP
NER	B-PERSON	O	O	O	O
SRL	B-ARG0	B-V	B-ARG1	I-ARG1	I-ARG1
POS	NNP	VBD	DT	NN	NN

Table 2: Example annotations for CHUNK, NER, SRL, and POS.

4 Experiments

Data We want to experiment with multiple loosely related NLP tasks, but also study performance across domains to make sure our architecture is not prone to overfitting. As testbed for our experiments, we therefore choose the OntoNotes 5.0 dataset (Weischedel et al., 2013), not only due to its high inter-annotator agreement (Hovy et al., 2006), but also because it enables us to analyze the generalization ability of our models across different tasks and domains. The OntoNotes dataset provides data annotated for an array of tasks across different languages and domains. We present experiments with the English portions of datasets, for which we show statistics in Table 1.[1]

Tasks In multi-task learning, one task is usually considered the main task, while other tasks are used as auxiliary tasks to improve performance on the main task. As main tasks, we use chunking (CHUNK), named entity recognition (NER), and a simplified version of semantic role labeling (SRL) where we only identify headwords[2], and pair them with part-of-speech tagging (POS) as an auxiliary task, following (Søgaard and Goldberg, 2016). Example annotations for each task can be found in Table 2.

Model We use a state-of-the-art BiLSTM-based sequence labeling model (Plank et al., 2016) as the building block of our model. The BiLSTM consists of 3 layers with a hidden dimension of 100. At every time step, the model receives as input the concatenation between the 64-dimensional embedding of a word and its character-level embedding produced by a Bi-LSTM over 100-dimensional character embeddings. Both word and character embeddings are randomly initialized. The output layer is an MLP with a dimensionality of 100. We initialize α parameters with a bias towards one source subspace for each direction and initialize β parameters with a bias towards the last layer[3]. We have found it most effective to apply the orthogonality constraint only to the weights associated with the LSTM inputs.

Training and Evaluation We train our models with stochastic gradient descent (SGD), an initial learning rate of 0.1, and learning rate decay[4]. During training, we randomly and uniformly sample from the data for each task. We perform early stopping with patience of 2 based on the main task and hyperparameter optimization on the in-domain development data of the newswire domain. We use the same hyperparameters for all comparison models across all domains. We train our models on each domain and evaluate them both on the in-domain test set (Table 3, top) as well as on the test sets of all other domains (Table 3, bottom) to evaluate their out-of-domain generalization ability[5].

Baseline Models As baselines, we compare against i) a single-task model only trained on chunking; ii) the low supervision model (Søgaard and Goldberg, 2016), which predicts the auxiliary task

[1]Note that not all sentences are annotated with all tasks.

[2]We do this to keep pre-processing for SRL minimal.

[3]We experimented with different initializations for α and β parameters and found these to work best.

[4]We use SGD as Søgaard and Goldberg (2016) also employed SGD. Adam yielded similar performance differences.

[5]Due to this set-up, our results are not directly comparable to the results in Søgaard and Goldberg (2016) who only train on the WSJ domain and use OntoNotes 4.0.

	System	bc	bn	mz	nw	pt	tc	wb	Avg
					In-domain results				
Baselines	Single task	90.80	92.20	91.97	92.76	97.13	89.84	92.95	92.52
	Hard parameter sharing	90.31	91.73	92.33	92.22	96.40	90.59	92.84	92.35
	Low supervision	90.95	91.70	92.37	93.40	96.87	90.93	93.82	92.86
	Cross-stitch network	91.40	92.49	92.59	93.52	96.99	**91.47**	94.00	93.21
Ours	Sluice network	**91.72**	**92.90**	**92.90**	**94.25**	**97.17**	90.99	**94.40**	**93.48**
					Out-of-domain results				
Baselines	Single task	85.95	87.73	86.81	84.29	90.91	84.55	73.36	84.80
	Hard parameter sharing	86.31	87.73	86.96	84.99	90.76	84.48	73.56	84.97
	Low supervision	86.53	88.39	87.15	85.02	90.19	84.48	73.24	85.00
	Cross-stitch network	87.13	88.40	87.67	85.37	91.65	**85.51**	73.97	85.67
Ours	Sluice network	**87.95**	**88.95**	**88.22**	**86.23**	**91.87**	85.32	**74.48**	**86.15**

Table 3: Accuracy scores on in-domain and out-of-domain test sets for chunking (main task) with POS tagging as auxiliary task for different target domains for baselines and Sluice networks. Out-of-domain results for each target domain are averages across the 6 remaining source domains. Average error reduction over single-task performance is 12.8% for in-domain; 8.9% for out-of-domain. In-domain error reduction over hard parameter sharing is 14.8%.

at the first layer; iii) an MTL model based on hard parameter sharing (Caruana, 1993); and iv) cross-stitch networks (Misra et al., 2016). We compare these against our complete sluice network with subspace constraints and learned α and β parameters. We implement all models in DyNet (Neubig et al., 2017) and make our code available at `http:anonymized.com`.

Our assumption is that MTL particularly helps to generalize to data whose distribution differs from the one seen during training. We thus first investigate how well sluice networks perform on in-domain and out-of-domain test data compared to state-of-the-art multi-task learning models by evaluating all models on chunking with POS tagging as auxiliary task.

Results We show results on in-domain and out-of-domain tests sets in Table 3. On average, sluice networks significantly outperform all other model architectures on both in-domain and out-of-domain data. Single task models and hard parameter sharing achieve the lowest results. We see that single task learning is comparatively most useful in the in-domain setting, where the distribution of the test data is the same as during training. On out-of-domain data, hard parameter sharing performs better, demonstrating the regularizing effect of MTL, which improves the model's generalization ability.

Both models are consistently outperformed by low supervision, which is only slightly better than hard parameter sharing in the out-of-domain setting. Cross-stitch networks provide another significant performance improvement. Finally, sluice networks perform best for all domains, except for the telephone conversation (tc) domain, where they are outperformed by cross-stitch networks. The performance boost is particularly significant for the out-of-domain setting, where sluice networks add more than 1 point in performance compared to hard parameter sharing and almost .5 compared to the strongest baseline on average, demonstrating that sluice networks are particularly useful to help a model generalize better.

In summary, this shows that our proposed model for learning which parts of multi-task models to share, with a small set of additional parameters to learn, can achieve significant and consistent improvements over strong baseline methods.

	System	nw (ID)	bc	bn	mz	pt	tc	wb	OOD Avg
Named entity recognition									
Baselines	Single task	95.04	93.42	93.81	93.25	94.29	94.27	92.52	93.59
	Hard parameter sharing	94.16	91.36	93.18	93.37	**95.17**	93.23	**92.99**	93.22
	Low supervision	94.94	91.97	93.69	92.83	94.26	93.51	92.51	93.13
	Cross-stitch network	95.09	92.39	93.79	93.05	94.14	93.60	92.59	93.26
Ours	Sluice network	**95.52**	**93.50**	**94.16**	**93.49**	93.61	**94.33**	92.48	**93.60**
Simplified semantic role labeling									
Baselines	Single task	97.41	**95.67**	95.24	95.86	95.28	98.27	97.82	96.36
	Hard parameter sharing	97.09	95.50	95.00	95.77	**95.57**	98.46	97.64	96.32
	Low supervision	97.26	95.57	95.09	95.89	95.50	98.68	97.79	96.42
	Cross-stitch network	97.32	95.44	95.14	95.82	**95.57**	**98.69**	97.67	96.39
Ours	Sluice network	**97.67**	95.64	**95.30**	**96.12**	95.07	98.61	**98.01**	**96.49**

Table 4: Test accuracy scores for different target domains with nw as source domain for named entity recognition (main task) and simplified semantic role labeling with POS tagging as auxiliary task for baselines and Sluice networks. ID: in-domain. OOD: out-of-domain.

System	CHUNK	NER	SRL	POS
Single task	**89.30**	94.18	96.64	88.62
Hard param.	88.30	94.12	**96.81**	89.07
Low super.	89.10	94.02	96.72	89.20
Sluice net	89.19	**94.32**	96.67	**89.46**

Table 5: All-tasks experiment: Test accuracy scores for each task with nw as source domain averaged across all target domains.

Performance across Tasks We now compare sluice nets across different combinations of main and auxiliary tasks. In particular, we evaluate them on NER with POS tagging as auxiliary task and simplified semantic role labeling with POS tagging as auxiliary task. We show results in Table 4. Sluice networks outperform the comparison models for both tasks on in-domain test data and successfully generalize to out-of-domain test data on average. They yield the best performance on 5 out of 7 domains and 4 out of 7 domains for NER and semantic role labeling.

Performance with all Tasks To the best of our knowledge, most of the existing work in multi-task learning for NLP employs two tasks, typically pairing a main task with an auxiliary task. Existing studies (Bingel and Søgaard, 2017; Martínez Alonso and Plank, 2017) also only evaluate pair-wise interactions between tasks. (Hashimoto et al., 2017) is the only model we are aware of that learns from multiple stand-alone NLP tasks, but uses a task-specific architecture to do so, while our model can be applied to any task combination. We use one sluice network to jointly learn our four tasks on the newswire domain and show results comparing it to the baseline models in Table 5[6].

Discussion For MTL with four tasks, the low-level POS tagging and simplified SRL tasks are the only ones that benefit from hard parameter sharing. This is consistent with results in Table 3 and previous work (Søgaard and Goldberg, 2016). These results highlight that hard parameter sharing by itself is not sufficient for doing effective multi-task learning with semantic tasks. We rather require task-specific layers that can be used to transform the shared, low-level representation into a form that is able to capture more fine-grained task-specific knowledge.

[6]For the low supervision model, we predict the two high-level tasks, CHUNK and NER at the final layer.

Task sharing	Layer sharing	bc	bn	mz	nw	pt	tc	wb	Avg
constant α (hard)	Concatenation	86.70	88.24	87.20	85.19	90.64	85.33	73.75	85.29
	Skip-connections ($\beta = 1$)	86.65	88.10	86.82	84.91	90.92	84.89	73.62	85.13
	Mixture (learned β)	86.59	88.03	87.19	85.12	90.99	84.90	73.48	85.19
learned α (soft)	Concatenation	87.37	88.94	87.99	86.02	**91.96**	**85.83**	74.28	86.05
	Skip-connections	87.08	88.62	87.74	85.77	91.92	85.81	74.04	85.85
	Mixture	87.10	88.61	87.71	85.44	91.61	85.55	74.09	85.73
	Mixture + subspaces	**87.95**	**88.95**	**88.22**	**86.23**	91.87	85.32	**74.48**	**86.15**

Table 6: Ablation analysis. Accuracy scores on out-of-domain (OOD) test sets for Chunking (main task) with POS tagging as auxiliary task for different target domains for different configurations of sluice networks. OOD scores for each target domain are averaged across the 6 remaining source domains.

With sluice networks, we are able to outperform the single task models for all tasks except chunking in the all-tasks setting. Generally, MTL with more than two stand-alone tasks is little explored in NLP and the best choices for hyperparameters such as the sampling ratio, when to start, and stop training for each task and whether to freeze or continue training already learned parameters remain to be discovered.

5 Analysis

Task Properties and Performance Bingel and Søgaard (2017) correlate meta-characteristics of task pairs and gains from hard parameter sharing across a large set of NLP task pairs. Inspired by this study, we correlate various meta-characteristics with error reductions and α, β values in sluice networks, as well as in hard parameter sharing. Most importantly, we find that a) multi-task learning gains, also in sluice networks, are higher when there is less training data, and b) sluice networks learn to share more when there is more variance in the training data (cross-task αs are higher, intra-task αs lower). Generally, α values at the inner layers correlate more highly with meta-characteristics than α values at the outer layers.

Ablation Analysis Different types of sharing may be more important than others. In order to analyze this, we perform an ablation analysis in Table 6. We investigate the impact of i) the α parameters; ii) the β parameters; and iii) the division into subspaces with an orthogonality penalty. We also evaluate whether concatenation of the outputs of each layer is a reasonable alternative to our mixture model.

Overall, we find that learnable α parameters are preferable over constant α parameters. Learned β parameters marginally outperform skip-connections in the hard parameter sharing setting, while skip-connections are competitive with learned β values in the learned α setting. In addition, modeling subspaces explicitly helps for almost all domains. To our knowledge, this is the first time that subspaces within individual LSTM layers have been shown to be beneficial[7]. Being able to effectively partition LSTM weights opens the way to research in inducing more structured neural network representations that encode task-specific priors. Finally, concatenation of layer outputs is a viable form to share information across layers as has also been demonstrated by recent models such as DenseNet (Huang et al., 2017).

Analysis of α values Figure 2 presents the final α weights in the sluice networks for Chunking, NER, and SRL, trained with newswire as training data. We see that a) for the low-level simplified SRL, there is more sharing at inner layers, which is in line with (Søgaard and Goldberg, 2016), while Chunking and NER also rely on the outer layer, and b) more information is shared from the more complex target tasks than vice versa.

Analysis of β values Inspecting the β values for the all-tasks sluice net in Table 5, we find that all tasks place little emphasis on the first layer, but prefer to aggregate their representations in different later layers of the model: The more semantic NER and

[7]Liu et al. (2017) induce subspaces between separate LSTM layers.

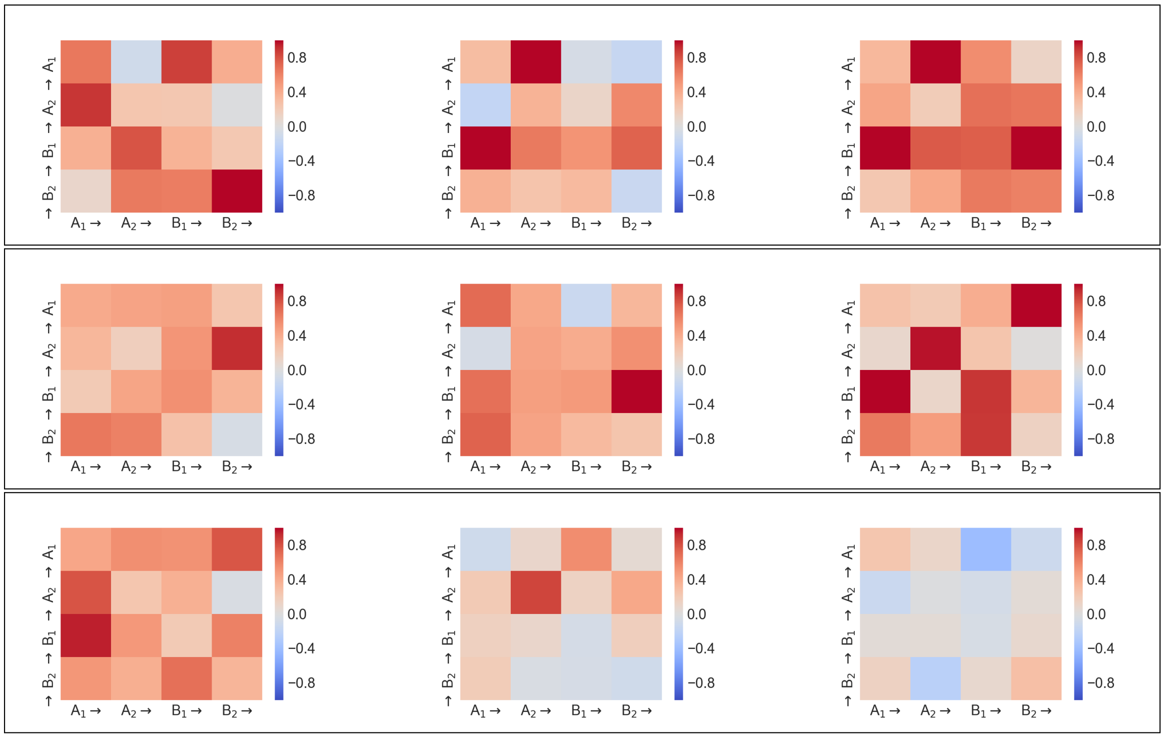

Figure 2: Heat maps of learned α parameters in trained sluice networks across (top to bottom): Chunking, NER, and SRL. We present inner, middle, and outer layer left to right.

chunking tasks use the second and third layer to a similar extent, while for POS tagging and simplified SRL the representation of one of the two later layers dominates the prediction.

Ability to Fit Noise Sluice networks can learn to disregard sharing completely, so we expect them to be as good as single-task networks to fit random noise, potentially even better. We verify this by computing a learning curve for random relabelings of 200 sentences annotated with syntactic chunking brackets, as well as 100 gold standard POS-annotated sentences. Figure 3 shows that hard parameter sharing, while learning faster because of the smoother loss surface in multi-task learning, is a good regularizer, confirming the findings in (Søgaard and Goldberg, 2016). The sluice network is even better at fitting noise than the single-task models. We believe the ability to fit noise in practical settings (on datasets of average size, with average hyper-parameters) is more informative than Rademacher complexities (Zhang et al., 2017) and we argue that this result suggests introducing additional inductive bias may be beneficial.

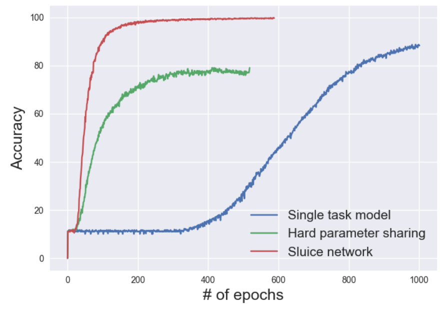

Figure 3: Random noise learning curves. Note that we only plot accuracies for hard parameter sharing and sluice networks until they plateau.

6 Related Work

In the context of deep neural networks, multi-task learning is often done with *hard or soft parameter sharing* of hidden layers. Hard parameter sharing was introduced by Caruana (1993). There, all hidden layers are shared between tasks which are projected into output layers specific to different tasks. This multi-task learning approach is easy to implement, reduces overfitting, but is only guaranteed to work for (certain types of) closely related tasks (Baxter, 2000; Maurer, 2007).

Peng and Dredze (2016) apply a variation of hard parameter sharing to multi-domain multi-task sequence tagging with a shared CRF layer and domain-specific projection layers. Yang et al. (2016) also use hard parameter sharing to jointly learn different sequence-tagging tasks (NER, POS tagging, Chunking) across languages. They also use word and character embeddings and share character embeddings in their model. Martínez Alonso and Plank (2017) explore a similar set-up, but sharing is limited to the initial layer. In all three papers, the amount of sharing between the networks is fixed in advance.

In soft parameter sharing, on the other hand, each task has separate parameters and separate hidden layers, as in our architecture, but the loss at the outer layer is regularized by the current distance between the models. In (Duong et al., 2015), for example, the loss is regularized by the L_2 distance between (selective parts of) the main and auxiliary models. Other regularization schemes used in multi-task learning include the ℓ_1/ℓ_2 group lasso (Argyriou et al., 2008) and the trace norm (Ji and Ye, 2009).

Selective Sharing Kumar and Daumé III (2012) and Maurer et al. (2013) enable selective sharing by allowing task predictors to select from sparse parameter bases for homogeneous tasks. Several authors have discussed which parts of the model to share for heterogeneous tasks. Søgaard and Goldberg (2016) perform experiments on which hidden layers to share in the context of hard parameter sharing with deep recurrent neural networks for sequence tagging. They show that low-level tasks, i.e. easy natural language processing tasks typically used for preprocessing such as part-of-speech tagging and named entity recognition, should be supervised at lower layers when used as auxiliary tasks.

Another line of work looks into separating the learned space into a private (i.e. task-specific) and shared space (Salzmann et al., 2010; Virtanen et al., 2011) to more explicitly capture the difference between task-specific and cross-task features. To enforce such behavior, constraints are enforced to prevent the models from duplicating information. Bousmalis et al. (2016) use shared and private encoders regularized with orthogonality and similarity constraints for domain adaptation for computer vision. Liu et al. (2017) use a similar technique for sentiment analysis.

In contrast to all the work mentioned above, we do not limit ourselves to a predefined way of sharing, but let the model learn which parts of the network to share using latent variables, the weights of which are learned in an end-to-end fashion.

The work most related to ours is (Misra et al., 2016), who also look into learning what to share in multi-task learning. However, they only consider a very small class of the architectures that are learnable in sluice networks. Specifically, they restrict themselves to learning *split architectures*. In such architectures, two n-layer networks share the innermost k layers with $0 \leq k \leq n$, and they learn k with a mechanism that is very similar to our α-values. Our work can be seen as a generalization of (Misra et al., 2016), including a more in-depth analysis of augmented works.

7 Conclusion

We introduced SLUICE NETWORKS, a framework for learning what to share in multi-task learning using trainable parameters. Our approach is a generalization of recent work, but goes well beyond this in enabling the network to learn selective sharing of layers, subspaces, and skip connections. In experiments with NLP task pairs in Ontonotes 5.0, we show up to 15% average error reduction over hard parameter sharing at only a 5–7% increase in training time. We provide an analysis of the ability of sluice networks to fit noise, as well as what properties are predictive of gains with sluice networks, seeing that the effect size correlates highly with label entropy, confirming previous findings for hard parameter sharing (Martínez Alonso and Plank, 2017; Bingel and Søgaard, 2017).

References

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2008. Convex Multi-Task Feature Learning. *Machine Learning*, 73(3):243–272.

Jonathan Baxter. 2000. A Model of Inductive Bias Learning. *JAIR*, 12:149–198.

Shai Ben-David and Reba Schuller. 2003. Exploiting Task Relatedness for Multiple Task Learning. In *Learning Theory and Kernel Machines*, pages 567–580. Springer.

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

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain Separation Networks. In *Proceedings of NIPS*.

Rich Caruana. 1993. Multitask Learning: A Knowledge-Based Source of Inductive Bias. In *Proceedings of ICML*.

Rich Caruana. 1998. Multitask Learning. In *Learning to Learn*, pages 95–133. Springer.

Ronan Collobert and Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of ICML*.

Hal Daumé III. 2007. Frustratingly Easy Domain Adaptation. In *Proceedings of ACL*.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low Resource Dependency Parsing: Cross-lingual Parameter Sharing in a Neural Network Parser. In *Proceedings of ACL*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. In *Proceedings of EMNLP*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of CVPR*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90 % Solution. In *Proceedings of NAACL-HLT*.

Gao Huang, Zhuang Liu, Kilian Q. Weinberger, and Laurens van der Maaten. 2017. Densely Connected Convolutional Networks. In *Proceedings of CVPR 2017*.

Laurent Jacob, Jean-Philippe Vert, Francis R Bach, and Jean-Philippe Vert. 2009. Clustered Multi-Task Learning: A Convex Formulation. In *Proceedings of NIPS*, pages 745–752.

Ali Jalali, Sujay Sanghavi, Chao Ruan, and Pradeep K Ravikumar. 2010. A Dirty Model for Multi-task Learning. In *Proceedings of NIPS*.

Shuiwang Ji and Jieping Ye. 2009. An Accelerated Gradient Method for Trace Norm Minimization. In *Proceedings of ICML*.

Abhishek Kumar and Hal Daumé III. 2012. Learning Task Grouping and Overlap in Multi-task Learning. In *Proceedings of ICML*, pages 1383–1390.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial Multi-task Learning for Text Classification. In *Proceedings of ACL*.

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

Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-paredes. 2013. Sparse coding for multitask and transfer learning. In *Proceedings of ICML*, volume 28, pages 343–351.

Andreas Maurer. 2007. Bounds for Linear Multi Task Learning. *JMLR*, 7:117–139.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-Stitch Networks for Multi-Task Learning. In *Proceedings of CVPR*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Nanyun Peng and Mark Dredze. 2016. Multi-task Multi-domain Representation Learning for Sequence Tagging. *CoRR*, abs/1608.02689.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *Proceedings of ACL*.

Mathieu Salzmann, Carl Henrik Ek, Raquel Urtasun, and Trevor Darrell. 2010. Factorized Orthogonal Latent Spaces. *JMLR*, 9:701–708.

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

Grzegorz Świrszcz and Aurélie C. Lozano. 2012. Multi-level Lasso for Sparse Multi-task Regression. In *Proceedings of ICML*.

Seppo Virtanen, Arto Klami, and Samuel Kaski. 2011. Bayesian CCA via Group Sparsity. In *Proceedings of ICML*.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2013. OntoNotes Release 5.0 LDC2013T19. *Linguistic Data Consortium*.

Yongxin Yang and Timothy M. Hospedales. 2017. Trace Norm Regularised Deep Multi-Task Learning. In *Proceedings of ICLR - Workshop Track*.

Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. 2016. Multi-Task Cross-Lingual Sequence Tagging from Scratch. *CoRR*, abs/1603.06270.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017. Understanding deep learning requires rethinking generalization. In *Proceedings of ICLR*.

Yang Zhou, Rong Jin, and Steven Hoi. 2010. Exclusive Lasso for Multi-task Feature Selection. In *Proceedings of AISTATS*.