Q-Learning Scheduler for Multi Task Learning Through the use of Histogram of Task Uncertainty

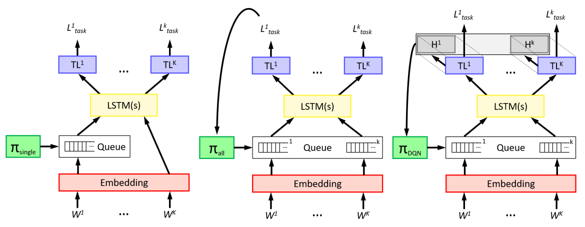
Kourosh Meshgi, Maryam Sadat Mirzaei & Satoshi Sekine
RIKEN Center for Advanced Intelligence Project (AIP)
Tokyo, Japan
{kourosh.meshgi, maryam.mirzaei, satoshi.sekine}@riken.jp

Abstract

Simultaneous training of a multi-task learning (MTL) network on different domains or tasks is not always straightforward. It could lead to inferior performance or generalization compared to the corresponding single-task networks. An effective training scheduling method is deemed necessary to maximize the benefits of multi-task learning. Traditional schedulers follow a heuristic or prefixed strategy, ignoring the relation of the tasks, their sample complexities, and the state of the emergent shared features. We proposed a deep Q-Learning Scheduler (QLS) that monitors the state of the tasks and the shared features using a novel histogram of task uncertainty, and through trial-and-error, learns an optimal policy for task scheduling. Extensive experiments on multi-domain and multi-task settings with various task difficulty profiles have been conducted, the proposed method is benchmarked against other schedulers, its superior performance has been demonstrated, and results are discussed.

1 Introduction

Multi-task learning aims to jointly improve the generalization of several classification and regression tasks. It does so by sharing the domain-centric information of each task, reducing trainable parameters, focusing attention on relevant features amid noisy or high dimensional data, regularizing other tasks, and exploiting the relations among tasks (Ruder, 2017; Zamir et al., 2018). The tasks can be defined as applying the same model on different data (also known as multi-domain learning) (Nam and Han, 2016; Liu et al., 2017a), or on various problems in a linear (Zamir et al., 2020) or hierarchical manner (e.g., named entity recognition, entity mention detection, and relation extraction in hierarchical MTL-HMTL (Sanh et al., 2019). Typical MTL is trained with mini-batches of every task intermittently, while the order of the training is usually uniform or related to the tasks' database size (Kiperwasser and Ballesteros, 2018). This simplistic approach leads to unnecessary computations, due to redundancy in tasks and samples (Lin et al., 2017), or due to the fact that some tasks are prerequisites for learning others (Ruder et al., 2017). Additionally, task imbalances deteriorate appropriate training because they lead to imbalances between back-propagated gradients (Chen et al., 2018b). Therefore, researchers have proposed methods to train the tasks and samples in a certain order, a process called "MTL Scheduling".



(a) Single Task Scheduler (b) All Tasks Scheduler (c) Proposed Scheduler

Figure 1: Different schedulers for multi-task learning **(a)** Single scheduler that adjusts the training priority of one task compared to others, **(b)** All scheduler that dynamically adjusts the relative importance of each task compared to others using key performance indicators (e.g., validation loss) of tasks, **(c)** Proposed scheduler that learns an optimal scheduling policy by employing deep Q-learning on task uncertainties. Similar scheduling effect can be achieved via adjusting learning rate and gradient manipulation, and training queue is depicted as an example of scheduling methods. In this figure, TL denotes task layer, π denotes the (trained) policy, and L^i_{task} indicates the loss of the specific task i.

Effective scheduling increases accuracy, reduces overfitting across multiple tasks, avoids catastrophic forgetting, improves the low-resource task accuracy. Meanwhile, it keeps the high resource task accuracy intact, provides extensive control over the training dynamics of MTL, and exploits task relations to learn required features for upsteam tasks. Early scheduling approaches include non-adaptive heuristics and fixed strategies to order the training. Such scheduling was applied to one target task (single scheduling, Figure 1(a)) or was used by scaling per-task learning rates (Jean et al., 2019). As these methods were not adequate to handle more complicated MTL systems (e.g., UberNet (Kokkinos, 2017)), more flexible and adaptive scheduling methods started to emerge. They use learning progress signals such as training loss (Kiperwasser and Ballesteros, 2018), validation loss (Jean et al., 2019), and uncertainty (Kendall et al., 2018) to tailor the schedule accordingly (Figure 1(b)). Advanced schedulers are expected to monitor task learning progress, emergent shared feature representation, and sample complexity of each task to be able to provide a suitable strategy for task orders. Additionally, when the underlying multi-task models learn to improve the performance of harder tasks, they may hit a plateau; as a result, simpler (or data-poor) tasks can be overtrained (overfitted). Needless to say that some tasks may be forgotten if the schedule is improper (catastrophic forgetting). Using a prefixed model to handle all these factors (model bias problem) without considering instantaneous feedback from the network (lack of temporal monitoring) is the main challenge for many of the current schedulers.

To address the challenges, we propose using reinforcement learning (RL) method to learn a potentially complex strategy. This allows for handling different states of the MTL training and avoids catastrophic forgetting. It also enables learning long-term temporal effects of task selection on the network's performance using the intrinsic delayed reward handling mechanism of RL. Moreover, it uncovers task relations with no explicit modeling (model-free), merely based on trial-and-error and receiving (delayed) feedback from the MTL training (Figure 1(c)). We use deep Q-Net (DQN) (Mnih et al., 2013) to map the state of the MTL to the desired actions (i.e., *which task to train on next?*), and propose the *histogram of task uncertainty* to describe the MTL state for the algorithm. Our contributions are:

- Introducing histogram of task uncertainty for the descriptive signal of the MTL;
- Proposing the use of deep Q-learning to learn the MTL scheduling to *(i)* handle temporal progress in MTL, *(ii)* provide sufficiently complex strategy for marginal cases, *(iii)* avoid catastrophic forgetting actively, and *(iv)* consider sample and task complexity;
- Extensive tests to investigate the performance and generalization of MTL's learned features;
- Experiments on multi-task and multi-domain learning problems with homo- or heterogeneous tasks, with various inter-task relations.

Note that in our experiments we focus on the performance of our Q-learning scheduler compared with other scheduling methods. We used LSTM instead of variations of transformers to make a fair comparison with other methods that used LSTM. In the future, we will incorporate QLS with transformers to benchmark the addition of such scheduling.

2 Related Works

Multi-Task Learning: To leverage from correlations of different tasks, MTL could be performed on tasks such as: those derived from different subsets of a shared data pool (Meyerson and Miikkulainen, 2018), adversarial tasks (Ganin and Lempitsky, 2015), auxiliary tasks which provide hints or attention for the main task (Yu and Jiang, 2016; Caruana, 1997), tasks arranged in an easy-to-hard hierarchy (Sanh et al., 2019), those which explicitly perform representation learning for a more complex application (Rei, 2017; Subramanian et al., 2018), or those which facilitate training for a quickly-plateauing main task (Bingel and Søgaard, 2017). Also, it can be helpful to learn the inter-task relations to enable efficient transfer learning or task grouping (Ruder et al., 2017; Bingel and Søgaard, 2017; Zamir et al., 2018; Standley et al., 2019).

In the hard parameter sharing architectures, shared parameters provide a global feature representation, while task-specific layers further process these features or provide a complimentary feature set for a specific task. MTL methods assume that learning easy tasks is the prerequisite for learning complex ones (Ruder, 2017), hence put tasks in hierarchies (Hashimoto et al., 2017; Sanh et al., 2019) or group similar tasks to form group-specific shared layers (Liu et al., 2017b).

Scheduling MTL: To train a deep neural network on a battery of tasks simultaneously, the tasks are sampled uniformly or in proportion to their dataset size (Jean et al., 2019). Since this may offer limited control over the performance trade-offs (e.g., accuracy vs. overfitting), task scheduling methods were proposed to improve the MTL's performance compared to single-task networks, static heuristics, and grid search methods. Thus, non-adaptive (fixed strategy) and adaptive methods were used, and various classes of scheduling emerged, such as:

Sample schedulers try to over-sample tasks with worse results compared to the baseline (Kiperwasser and Ballesteros, 2018) or down-weight easier samples to focus on harder ones for training (Lin et al., 2017). Yet, such strategies fail if a task is highly over-sampled or datasets are imbalanced.

Task difficulty schedulers are usually built upon the notion of curriculum learning that favors smaller and easier tasks to learn first (Pentina et al., 2015), aligned with the training of natural intelligence in human babies. This idea falls into two forms, task hierarchies (e.g., (Sanh et al., 2019)) and task prioritization (Pentina et al., 2015; Graves et al., 2017; Zaremba and Sutskever, 2014). In the latter case, if the data of the tasks are coming from significantly different distributions (e.g., domain adaptation (Luo et al., 2017; Glorot et al., 2011; Tzeng et al., 2015)), the assumptions of curriculum learning do not hold (Bengio et al., 2009), and prefixed schedulers might not be effective.

Task weighting is another approach for scheduling problem. MTL is sensitive to the task weights (Kendall et al., 2018). These weights scale the loss term of each task in the total loss of the network. Static task weights can be selected by hyperparameter tuning (Kokkinos, 2017; Sermanet et al., 2013), yet this approach is suboptimal in the presence of less important or redundant tasks/samples (Lin et al., 2017). Non-adaptive weighting schemes fetch tasks intermittently early in training and gradually weigh more on a specific task (Jean et al., 2019). Adaptive schedulers dynamically prioritize different tasks by monitoring measures such as performance (Jean et al., 2019) and homoscedastic uncertainty (Kendall et al., 2018). Another method is to dynamically tune the gradient magnitudes (Chen et al., 2018b). Here, we use Q-learning scheduler which can dynamically schedule the learning of long-term temporal effects of task selection. As a result of scheduling, QLS selects the task to draw samples for the next training episodes.

3 Multi-task Classification

We selected text classification, in which a document needs to be assigned to a set of classes. The solutions range from hand-crafting good features to be used in convolutional neural networks (NNs) for word-level (Kim, 2014) and character-level encoding (Zhang et al., 2015), recurrent NNs (Liu et al., 2016) and convolutional recurrent NNs (Lai et al., 2015). Here, we address the problem of text classification using long-short term memory networks (LSTM) with a variant explored in (Jozefowicz et al., 2015) to facilitate further analysis of the effects of the proposed method in the representation. Generally, this method can be applied to any encoder-decoder-based NLP task that uses LSTM as the encoder. The text sequence of words $\mathbf{w} = \{w_1, w_2, \ldots, w_T\}$ is converted to a sequence of word embeddings $\mathbf{x}_i$ and is given to an LSTM layer. Each cell of LSTM layer at time t, includes input, forget and output gates, a memory and a hidden state $\mathbf{h}_t$. The LSTM memory chain updates as

$$\mathbf{h}_t = \mathrm{LSTM}\left(\mathbf{h}_{t-1}, \mathbf{x}_t, \theta_p\right) \tag{1}$$

where the output of the last unit $\mathbf{h}_T$ represents the whole sequence, and θ_p encapsulates the weights and biases of the LSTM. This is then fed to the task-specific output layers. The network is then trained on a training corpus with N samples $(\mathbf{w}_i, y_i)$ using cross-entropy loss function

$$L(\hat{y}, y) = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_i^j \log\left(\hat{y}_i^j\right) \tag{2}$$

where y_i^j is the groundtruth in $\{1..C\}$ and $\hat{y}_i^j$ is the predicted probability of label j for document i. By exploiting commonalities and differences among tasks, multi-task learning aims to improve the learning efficiency and prediction accuracy for all tasks by learning from them in parallel. To this end, a learner shares some of its parameters between tasks while keeping some of them specific to each task. Considering our baseline classifier, the shared features are the hidden states of the LSTM at the end of the input sequence. There are several ways to implement the MTL using this baseline for classification. The most popular idea is to use a fully-shared model, in which all tasks are using the extracted features of the shared LSTM layer, and then differentiate using a final task layer.

We denote different datasets D_k as datasets with N_k examples for task k, $D_k = \left\{\left(w_i^{\langle k\rangle}, y_i^{\langle k\rangle}\right)\right\}_{i=1}^{N_k}$. Given task k, the final task-specific softmax layer for classification, converts the shared feature $\mathbf{h}_T^{(k)}$ into probability distribution $\hat{y}^{(k)}$. The parameters of the network are trained by minimizing the cross-entropy of true distribution of the task $y^{(k)}$ and the predicted distribution $\hat{y}^{(k)}$, using the loss

$$L_{task} = \sum_{k=1}^{K} \alpha_k L\left(\hat{y}^{\langle k\rangle}, y^{\langle k\rangle}\right). \tag{3}$$

Here, α_k is the importance of each task k and $L(\hat{y}, y)$ is defined in eq(2). A scheduler, when applied to this MTL frameworks, adopts one of the following modifications: change the task weight (α_k), modify its gradient in the back-propagation, re-weight the samples ($\mathbf{x}_t$), or select a task to draw samples for the next training episodes. Here, for simplicity, we select the latter case to focus more on the idea. Yet, the extension of the proposed idea to all different approaches is straightforward.

4 Proposed Method

4.1 Progress Signal for Learning Strategy

Graves et al. (Graves et al., 2017) employ accuracy as a learning progress signal to find a policy for task curriculum learning (Oudeyer et al., 2007). A syllabus of curriculum learning is selected using this learning progress signal to maximize the overall training progress. Progress signals are typically used in RL problems as reward signals to encourage exploration (Schmidhuber, 1991; Itti and Baldi, 2006; Houthooft et al., 2016). Similarly, Routing Networks (Rosenbaum et al., 2017) select different network submodules, based on the task and rewards via a multi-agent formulation. An ambitious idea is to train an agent that is capable of designing the entire network architecture in neural architecture search (Zoph and Le, 2016) using accuracy as the signal. DTP (Guo et al., 2018) scheduler uses prediction gain (Bellemare et al., 2016) to dynamically compute task weights/priority during training.

Here, we use overall task uncertainty and validation loss as the progress signal of the network. Together, these metrics give a comprehensive view of the state of the MTL.

4.2 Proposed Histogram of Task Uncertainty

Task uncertainty measures what the model does not know or what cannot be inferred from the data (Kendall and Gal, 2017). In MTL, we need to learn multiple tasks simultaneously, while the uncertainty of each task varies. In a fully-shared MTL setting, each task contributes to the loss function based on the errors it make, and since one task is being trained at a time, minimizing this error may negatively change the shared parameters for other tasks. Using uncertainty instead of task accuracy provides an additional signal to train the model.The classifiers suffer from uncertainty when choosing the label. However, adding model uncertainty to the loss helps classifiers to make more determined decisions.This helps by reflecting the internal state of the classifiers. The model uncertainty can be obtained via Monte Carlo dropout sampling (Kendall et al., 2015), as a function of the samples' variance to be used as an estimation of the error (Kendall and Cipolla, 2016). Another way is to compute the standard deviation over the softmax outputs and average them over all classes to obtain a single value (Kampffmeyer et al., 2016). Moreover, Kendall et al. (2018) calculated homoscedastic uncertainty (the uncertainty of the entire task itself not dependent on input data) and used this to learn a weighting for each loss term in a multi-task setting.

To calculate the uncertainty of a multi-class classifier, uncertainty sampling methods could be applied. Thus, we proposed the uncertainty loss term as using margin uncertainty (Scheffer et al., 2001)

$$\zeta_M = 1 - P_k\left(\hat{y}^{(1)}|x\right) + P_k\left(\hat{y}^{(2)}|x\right), \qquad (4)$$

where $\hat{y}^{(j)}$ $(j = 1, 2)$ is the label with jth largest predicted probability. For each task k all validation data are given to the MTL, and their label uncertainty is calculated using margin uncertainty. The final histogram of task uncertainties, $\mathcal{H}_t^{(k)}$, is formed by concatenating all histograms.

4.3 Q-Learning Scheduler

In our method, QLS monitors the state of the tasks to measure the task uncertainty, generates the task uncertainty histogram, and then uses Q learning to schedule tasks. We formulate a Q-learning agent to adjust the histogram of task uncertainty for the proposed MTL scheduler. At time t, the agent takes an action a_t based on the state S_t of the MTL environment, and the environment gives the reward $r(S_t, a_t)$ and updates its state to S_{t+1}. The agent chooses its action w.r.t its policy $\pi(a_t|S_t)$ to maximize the cumulative reward $R_t = \sum_{i=t}^{T} \gamma^{i-t} r(S_i, a_i)$. Here, $0 < \gamma \leq 1$ is the discount factor to weigh more on earlier rewards. Q-learning calculates Q-values, which is the expected max scores for each action a_t in state S_t, as

$$Q(S_t, a_t) = r(S_t, a_t) + \gamma Q(S_{t+1}, a_{t+1}) \qquad (5)$$

State: The state $S_t \in \mathbf{S}_t$ of the environment is explained using the concatenation of n_b-bin histogram of uncertainty measurements of the main classifier for all samples $\mathbf{x}_t$, for all tasks. To eliminate the effect of the stochastic sampling on the uncertainty histogram, a deterministic sampling approach is used, which obtains the batch hard samples (Hermans et al., 2017) from the validation set of each task. The histogram of task uncertainty, $\mathcal{H}_t^{(k)}$, forms the input of the DQN network.

Action: Actions $a_t \in \mathbf{A}_t$ are K one-hot vectors, each indicating the task that is to be trained next.

Reward: During training time, the reward is defined as the $-L_{task}$ on all of the validation samples which includes the summation of losses of all tasks (eq(3)).If the average accuracy of the MTL drops under 95% of the single-task network, a big punishment (manually set to -10) is fed back to the scheduling learner agent to punish the use of chosen policy.

Policy: During the training time, we use Boltzmann-Gumbel exploration (Cesa-Bianchi et al., 2017) to exploit all the information present in the estimated Q-values with an additional temperature parameter, which is annealed over time. This parameter controls the spread of the softmax distribution so that at the start of the training, the equal chance is assigned to each action while actions are sparsely distributed by the end of the training.

4.4 Implementation Details

Our proposed method, QSL-MTL, used LSTM of length 128, GloVe (Pennington et al., 2014) word embedding (300d version on 840B Common Crawl data), and Xavier initialization for the parameters. The mini-batch size is set to 16, including samples of the same task. Other than these, we follow the training procedure and hyper-parameter settingin (Søgaard and Goldberg, 2016). The Q-learning parameters are then set to fixed values of $n_b = 20$ and $\gamma = 0.99$, and the Q-values are randomly initialized.

We used different uncertainty measures such as using least confidence (Settles and Craven, 2008), margin (Scheffer et al., 2001), and Shanon's entropy to calculate the histogram of uncertainty. We also experimented with accuracy for the initial phase to find the best performance according to the preliminary results. We used 10K iterations for our initial testing to explore the best practice and found the margin loss has the best performance.

The proposed scheduler is trained on the training data of all tasks. The rest ofhyper-parameters are tuned over the validation sets. For each of the 1M training episodes, we randomly sample a minibatch from one of K tasks, and reset the MTL modelevery 10K training episode to enable exploring other cases in the MTL. We used GeForce GTX 1080 GPU. During run-time, the scheduler greedily selects the action a_t^* which yields the highest expected reward, $a_t^* = \text{argmax}_{a_t' \in \mathbf{A}_t} Q(S_t, a_t')$. In experiments, we run our system three times and report the average. Note that unlike other methods that randomly fill minibatches with samples, our proposed method learns the policy for sample selection. Although training time takes longer, our method saves a lot of time by finding the best learning policy by itself using q-learning hence eliminates human feature selection procedure and is able to discover surprisingly good features which may not be straightforward for human.

5 Experiment

Our proposed scheduling method is benchmarked under three different scenarios: *(i)* A multi-domain setting, in which a similar task is performed on different datasets, *(ii)* a simple multi-task setting, where the network does three different but slightly related tasks, and *(iii)* a complex multi-task setting in which task relations are complex (e.g., one task can be a prerequisite for another, improving one may hamper the performance of another, etc.). We have conducted an extensive analysis of the first task to clarify our proposed method's internal mechanism and advantages.

5.1 Multiple Domains

In this experiment, we consider text classification as the learning task and 16 different datasets as domains of the task. Each domain consists of around 2000 comments about a class of products labeled as positive or negative reviews and 2000 unlabeled comments. We have applied various MTL scheduling methods on the fully-shared MTL (hereafter, the baseline) and compared the performance of our proposed method to the competitors and their performance on unseen data.

Dataset: Fourteen product review datasets for different products from (Blitzer et al., 2007) have been obtained as domains, with their labels as positive (4+ stars) or negative (2- stars), omitting the borderline 3-star comments. IMDB and MR movie review datasets from (Maas et al., 2011) and (Pang and Lee, 2005) with binary labels (subjective/objective and positive/negative) are also used as two additional domains. Table 1 shows domain statistics.

Competitor Models: We compared our algorithm with several scheduling strategies that are implemented on top of Fully-Shared MTL with a pre-trained word embedding and shared LSTM layers

Domain	Books	Elecs	DVD	Kitchen	Apparel	Camera	Health	Music	Toys	Video	Baby	Mags	Soft	Sports	IMDB	MR
Train	⋆	1398	⋆	⋆	⋆	1397	⋆	⋆	⋆	⋆	1300	1370	1315	⋆	⋆	⋆
Dev								— All 200 —							200	200
Test								— All 400 —							400	400
Len	159	101	173	89	57	130	81	136	90	156	104	117	129	94	269	21
Vocab	62K	30K	69K	28K	21K	26K	26K	60K	28K	57K	26K	30K	26K	30K	44K	12K

Table 1: Datasets' statistics (i.e., domains) for multi-domain text classification experiment. (⋆ = 1400)

(baseline). *Uniform*: All tasks have the same importance from beginning to the end of training; *Hand Crafted*: Tasks received a fixed importance coefficient for all training obtained by grid search; *Random*: A random task is selected for the next training episode; *Greedy*: The task with the highest loss is selected for the next training episode; *Loss Exponentiation*: Loss outputs are magnified by the power of 1.15 (found by a grid search). In this way, larger losses (for the tasks needing more changes in the shared space) are magnified; *Homoscedastic Uncertainty* (Kendall et al., 2018): Calculates the task uncertainty using loss magnitude and use it to weigh different tasks; *Self Paced* (Li et al., 2017): Introduces task weights as learnable parameters and employs a regularization that favors training on easy tasks earlier in the training process; *Focal Loss* (Lin et al., 2017): Sample-level scheduler that down-weights easier samples and focuses on hard samples during training; *DTP* (Guo et al., 2018): Uses learning progress signals to automatically compute a priority level at both a task-level and example-level.; *Grad Norm* (Chen et al., 2018b): Scales task gradients based on the magnitude of the gradients and training losses; *Adaptive* (Jean et al., 2019): Oversamples tasks with poorer results compared to their baseline; *QLS*: Our proposed fully-trained Q-learning-based scheduling method.

Task-Specific Output Layer: The obtained shared representation is fed to the task-specific output classifiers composed of a fully connected layer followed by a softmax layer to predict the labels.

Performance Evaluation and Discussion: We perform the multi-domain learning on all 16 tasks to compare the task-specific and overall performance of the proposed method. All schedulers are added on top of untrained FS-MTL (baseline), and the training is governed by the scheduling strategy. Based on Table 2, in most of the cases, our learned strategy outperforms other strategies. An in-depth analysis revealed that in the early stages of the training, for instance, the performance of the `Kitchen` domain was higher, while other domains were still trying to improve their performances. Additionally, hand-crafted domain weights are working well for most of the domains. Yet, these weights are fixed, resulting in the suboptimal performance of this strategy. The uncertainty weighting by (Kendall et al., 2018) also ignores some categories since the early emergent features in the shared space are suboptimal for the task. Although these measures keep the tasks' homoscedastic uncertainty low, they fail to guarantee high performance. This finding calls for better uncertainty measures in such an approach. One of the shortcomings of the GradNorm algorithm was observed in cases that a very noisy minibatch from a task was selected. By largely redirecting the gradients to the corresponding task, GradNorm magnifies the label noise in the training of the task, causing some confusion in updating the feature space. Moreover, we observed that by using Q-Learning, our method *(i)* enables the discovery of more latent features (especially when two or more tasks have a mutual tacit feature that assists those tasks to have better overall performance), *(ii)* easily switches between hard and easy tasks periodically to learn the policies and *(iii)* finds longer sequences as features that work well in the run phase. We also found two groups of mistakes by our model: *(i)* sentences with complicated structures such as when two negative words are separated by two or more words and *(ii)* sentences that require reasoning or external references (e.g., to pop culture) that conveys a particular sentiment, analogies (e.g., "The actors really are made of cardboard") or other types of inferences, out of the dataset's scope.

Shared Knowledge Transfer: One of the reasons for using MTL methods is to obtain a better-shared representation between tasks that cancels out the systematic noise of each individual task and provides a generalizable feature set that performs well out-of-the-box on unseen data. We hypothesize that by properly scheduling the MTL, more generalizable and useful features emerge early in the feature space. Therefore, the training procedure focuses more on improving such features rather than using some inefficient or suboptimal features and discarding them later. To test this hypothesis, we perform a leave-one-out experiment in which the proposed classifier is trained on 15 tasks and tested on one task which was excluded from training (e.g., we train on all tasks/categories except ϕ`(Book)` then we test on ϕ`(Book)` category, we then do the

Domain	Uniform (baseline)	Hand Crafted	Random	Greedy	Loss Exponen.	Homos. Unc.	Self Paced	Focal Loss	DTP	Grad Norm	Adaptive	**QLS** (ours)
Books	82.5	89.1	83.3	83.0	83.4	87.8	88.1	87.0	90.3	89.3	89.1	90.4
Electronics	85.7	91.1	87.2	86.0	86.6	90.7	90.7	90.6	92.8	92.4	92.3	92.8
DVD	83.5	89.8	84.4	83.8	84.4	88.6	88.4	88.8	90.9	90.2	90.0	90.9
Kitchen	86.0	93.1	86.8	86.0	86.9	90.1	91.6	93.1	87.2	92.7	92.6	93.2
Apparel	84.5	88.8	85.8	85.0	85.3	89.1	89.0	88.4	90.9	91.2	90.9	91.3
Camera	86.5	91.3	89.4	86.8	87.3	91.4	91.3	90.8	93.3	93.2	92.9	93.2
Health	88.0	91.9	88.4	88.1	88.9	91.6	91.6	91.7	90.3	92.6	92.5	92.6
Music	81.2	87.8	81.9	81.4	82.0	86.6	86.5	86.9	89.0	87.9	87.6	89.1
Toys	84.5	90.4	85.0	84.7	85.4	89.6	89.1	88.8	91.8	91.4	91.2	92.2
Video	83.7	89.9	84.6	83.7	84.6	88.9	88.6	89.4	91.3	90.5	90.2	91.9
Baby	88.0	90.0	89.1	88.2	88.9	92.0	91.9	89.4	93.1	94.7	94.7	94.6
Magazines	92.5	92.6	93.1	92.5	93.4	92.2	91.9	90.7	93.0	94.0	93.6	94.2
Software	86.2	90.1	87.3	86.3	87.1	90.8	90.3	88.2	92.6	92.9	92.6	93.1
Sports	85.5	88.6	86.7	85.8	86.3	89.8	89.9	86.8	91.3	92.2	91.9	92.2
IMDB	82.5	89.7	83.5	82.8	83.4	87.9	87.9	89.0	90.5	89.2	89.1	91.3
MR	74.7	78.8	78.7	74.7	75.6	79.3	79.6	76.9	81.0	81.3	81.3	81.4
AVG	84.7	89.6	86.0	84.9	85.6	89.2	89.2	88.5	90.6	91.0	90.8	91.5

Table 2: Accuracy of different scheduling methods on 16 domains, compared to its Fully-Shared-MTL baseline (First, second, and third ranks). Our QLS method outperforms others in almost all of tasks.

same for the next category). We freeze the weights of the trained shared model, perform 5-fold cross-validation on the left-out task, and report the result in Table 3. For each unseen domain, new task layers are made on top of the shared feature space. The task layer israndomly initialized, and trained on a new domain.

5.2 Multiple Tasks with Simple Relations

This experiment considers a heterogeneous multi-task learning scenario in which three different tasks (part-of-speech tagging, chunking, and named entity recognition) on various datasets are considered. Despite their differences, these tasks are related, but none of them could benefit from the output of others. We trained FS-MTL with different strategies and compared their performance on these different tasks (Table 5). We excluded pre-training from our model to provide a fair comparison (as in the modern Transformer sense).

Domain	Hand Craft	Loss Exp.	Homos Unc.	Self Paced	Focal Loss	DTP	Grad Norm	Adapt.	**QLS** (ours)
ϕ (Books)	86.3	81.8	85.9	82.2	81.7	86.5	86.3	86.4	86.6
ϕ (Elec.)	86.2	83.6	86.1	85.0	84.1	86.4	86.1	86.1	86.3
ϕ (DVD)	86.8	84.5	86.7	85.6	85.0	86.6	87.0	86.8	86.9
ϕ (Kitchen)	86.5	84.1	86.4	84.7	84.8	86.7	86.7	86.5	86.6
ϕ (Apparel)	86.3	84.7	86.2	85.1	84.7	86.2	86.3	86.2	86.3
ϕ (Camera)	87.3	86.2	87.3	86.9	86.2	87.3	87.1	87.1	87.3
ϕ (Health)	89.0	84.4	88.7	85.3	85.1	89.0	89.1	89.0	89.3
ϕ (Music)	85.6	79.7	85.0	80.2	80.2	85.9	85.9	85.9	86.1
ϕ (Toys)	86.3	83.7	86.1	84.1	84.1	85.6	86.3	86.0	86.4
ϕ (Video)	86.0	85.0	86.0	86.1	85.7	85.7	85.9	85.8	85.9
ϕ (Baby)	86.1	82.9	85.9	83.8	83.1	86.3	86.2	86.2	86.3
ϕ (Mags)	90.5	88.8	90.5	89.9	89.4	90.5	90.3	90.4	90.6
ϕ (Soft)	87.3	84.0	87.0	84.0	84.0	86.6	87.5	87.2	87.6
ϕ (Sports)	85.9	83.2	85.7	84.2	83.8	86.1	86.0	85.9	86.0
ϕ (IMDB)	87.7	86.8	87.7	87.5	87.1	87.5	87.6	87.5	87.6
ϕ (MR)	75.7	73.7	75.6	74.3	74.0	75.7	75.4	75.4	75.8
ϕ (AVG)	86.2	83.5	86.0	84.3	83.9	86.2	86.2	86.2	86.4

Table 3: Accuracy of fully shared representation learned with different strategies on all-but-one domains tested on the remaining unseen domain. ϕ(DOMAIN) means we transfer the knowledge of other 15 tasks to the target DOMAIN. Using the proposed scheduling of all tasks while training, we improved overall accuracy of MTL classifier on unseen data by 2.2% compared to baseline.

Task-Specific Output Layer: Inspired by (Ma and Hovy, 2016), obtained shared representation is fed to a conditional random field (Lafferty et al., 2001) for sequence tagging. Baseline has a pre-trained embedding, fully-shared LSTM(s), and a CRF.

Dataset: For sequence tagging task, we use Wall Street Journal (WSJ) subset of Penn Treebank (Marcus et al., 1993), CoNLL 2000 chunking, and CoNLL 2003 English NER dataset as in Table 4.

Datasets	Task	Train	Dev	Test
WSJ	POS Tagging	912,344	131,768	129,654
CoNLL 2000	Chunking	211,727	-	47,377
CoNLL 2003	NER	204,567	51,578	46,666

Table 4: Statistics of datasets for multi-task sequence tagging experiment.

Competitor Models: We compare our method with Huang et al. (2015) that uses a BiLSTM encoding and CRF output layer, text classifier of (Collobert et al., 2011), and multi-task text classifier with Meta-LSTM (Chen et al., 2018a). We trained the baseline MTL with different strategies such as DTP (Guo et al., 2018), Grad Norm (Chen et al., 2018b), and Adaptive scheduler (Jean et al., 2019) that performed best in the previous task. We also used Hand Crafted (worked well by finding fixed task weights) and uncertainty-based loss weighting (Kendall et al., 2018) to compare with our QLS.

Results and Discussion: Table 5 shows that our model consistently outperforms others. It is robust and has good generalization among related tasks. However, the task prioritization of Homoscedastic uncertainty and Grad Norm works well for some tasks but not others. One reason lies in the differences in task complexities: the emergent features are not always successful in handling the complexity of all tasks. Grad Norm aggressively decreases the relative weight of Chunking loss leading to a higher error rate in this task, and Homoscedastic uncertainty favors NER that made fewer mistakes early in training. While DTP works well, adaptive scheduling shows a mediocre performance.

5.3 Multiple Tasks with Complex Relations

In this experiment, we use different tasks (two easy and one complex [translation]) to investigate the effect of scheduling on the target task. We selected neural machine translation (NMT) as the target task and chose POS tagging and Parsing as the other tasks in MTL. As Table 6 shows, our method outperforms other static and dynamic scheduling methods. Compared to the baseline single-task NMT (with 19.30 BLEU score), our scheduled ML gains +2.6 points improvement in BLEU points.

Task-Specific Output Layer: NMT has an LSTM encoder and a seq2seq decoder (Sutskever et al., 2014; Bahdanau et al., 2014). Here, to be consistent with the literature, we perform POS tagging as a translation between source language and sequence of POS tags, similar to (Niehues and Cho, 2017). We also used the decoder in (Kiperwasser and Ballesteros, 2018) for dependency parsing.

Dataset: For translation setting, we use WMT'14 parallel corpus (Buck et al., 2014) including 4.5M training sentence pairs, 3000 sentences of *newstest2013* as the development set, and *newstest2014* for test set. English POS tagging, dependency heads and labels are from the Penn tree-bank with Stanford Dependencies (Training:02-21, Dev:22, Test:23) and German ones from TIGER tree-bank

	Chunking (CoNLL2000)	NER (CoNLL2003)	POS Tagging (WSJ)
BiLSTM+CRF	93.67	89.91	97.25
Meta-BiLSTM+CRF	93.71	90.08	97.30
(Collobert et al., 2011)	94.32	89.59	97.29
Meta-MTL + CRF	95.11	90.72	97.45
FS-MTL + CRF⋆	94.18	89.99	97.14
+ Hand Crafted	94.16	89.42	97.33
+ Homos. Unc.	95.05	90.24	97.42
+ DTP	95.46	90.73	97.38
+ Grad Norm	95.07	90.30	97.38
+ Adaptive	94.97	90.18	97.46
+ **QLS** (ours)	95.91	91.02	97.46

Table 5: Accuracy rates of the models for chunking and NER tasks using F1-score (%) and for POS tagging using Accuracy (%). Our QLS method outperforms others in most of the tasks. (⋆: baseline)

	Prior. Easy	Prior. Hard	Hand Crafted	Sig-moid	Expo-nential	Homo. Unc.	DTP	Grad Norm	Ada.	**QLS** (ours)
MT+POS	18.9	19.1	20.9	19.2	20.0	19.2	21.3	20.5	20.9	21.7
MT+Par.	18.7	18.6	20.5	19.1	18.9	18.9	18.9	20.4	21.1	21.4
MT+POS+Par.	19.0	18.3	20.9	19.3	18.0	19.0	20.8	20.7	21.5	21.9

Table 6: BLEU score of target machine translation with POS tagging and parsing (written as par.) as auxiliary tasks in multi-task framework trained with different scheduling strategies. The BLEU score of baseline Neural MT system (without auxiliary tasks) is 19.30. QLS shows superior performance among schedulers.

(Hajič et al., 2009). Translation quality is measured with case-sensitive BLEU (Papineni et al., 2002).

Competitor Models: We have included three types of schedulers: *(i)* all-task schedulers such as DTP (Guo et al., 2018), Grad Norm (Chen et al., 2018b), Adaptive (Jean et al., 2019), Homoscedastic Uncertainty (Kendall et al., 2018), and ours; *(ii)* one-task scheduler which tunes the weight of the target task over training episodes while keeping the rest of the weights intact; and *(iii)* constant schedules that assign each task a fixed weight for the entire training. Second category contains *exponential schedule* and *sigmoid schedule* (Kiperwasser and Ballesteros, 2018), and third category includes *prioritize hard/easy* strategies that assign the weight of 0.98 to the hardest/easiest task and 0.01 to others.

Results and Discussion: We selected NMT that is significantly harder than other tasks. Thus, the syntax representations learned by POS and Parsing tasks are required to process the input better and generate more meaningful sentences. As the table shows, merely plugging different tasks in the MTL framework does not guarantee better results as *(i)* some task combinations are not compatible (e.g., NMT and Parsing), which is aligned with the findings of (Zamir et al., 2020), and *(ii)* unified task weights can be damaging to the overall performance of a target model, while it might increase the overall performance of all tasks. It can be seen that even a carefully selected task's weighting (although static) could significantly improve this situation. Another observation is that single task schedulers are not always successful (only NMT + POS + Exponential scheduler improves the results in Table 6). DTP, Grad Norm, and homoscedastic uncertainty weighting suppress the performance of the NMT task, either because of its high initial error rate or due to favoring easier tasks that show better improvement during training. Among these, DTP sometimes covers difficult tasks better in the cost of performance of the overall tasks, depending on its KPI. On the other hand, adaptive scheduling handles catastrophic forgetting better and improves overall performance. However, we see that a tailored strategy (that our RL scheduler achieved through numerous trial-and-errors) works a lot better in handling such task difficulty imbalance. Yet, our method may not perform the best in handling balanced tasks or complex tasks that can benefit from sufficient amount of related easy ones.

6 Conclusion

We augment the fully-shared MTL framework with a reinforcement-learning-based scheduling scheme that obtains an optimal scheduling policy for tasks through trial and error. The scheduler detects the task states through a novel state definition: histogram of task uncertainty. It adjusts the scheduling policy to improve the training and validation accuracy of the MTL, enhances generalization of the emergent shared features, and handles different relationships among tasks. The proposed method is also capable of leveraging unlabeled data, obtaining highly-nonlinear strategies, and tackling different sources of task uncertainty.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying count-based exploration and intrinsic motivation. In *Advances in neural information processing systems*, pages 1471–1479.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *ACL'15*, pages 164–169.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 440–447.

Christian Buck, Kenneth Heafield, and Bas Van Ooyen. 2014. N-gram counts and language models from the common crawl. In *LREC*, volume 2, page 4. Citeseer.

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

Nicolò Cesa-Bianchi, Claudio Gentile, Gábor Lugosi, and Gergely Neu. 2017. Boltzmann exploration done right. In *NIPS'17*, pages 6284–6293.

Junkun Chen, Xipeng Qiu, Pengfei Liu, and Xuanjing Huang. 2018a. Meta multi-task learning for sequence modeling. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018b. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, pages 794–803.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *ICML'15*, pages 1180–1189.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*.

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. 2017. Automated curriculum learning for neural networks. *arXiv preprint arXiv:1704.03003*.

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

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štepánek, et al. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages.

Kazuma Hashimoto, Yoshimasa Tsuruoka, Richard Socher, et al. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *EMNLP'17*, pages 1923–1933.

Alexander Hermans, Lucas Beyer, and Bastian Leibe. 2017. In defense of the triplet loss for person reidentification. *arXiv preprint arXiv:1703.07737*.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. 2016. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Laurent Itti and Pierre F Baldi. 2006. Bayesian surprise attracts human attention. In *Advances in neural information processing systems*, pages 547–554.

Sébastien Jean, Orhan Firat, and Melvin Johnson. 2019. Adaptive scheduling for multi-task learning. *arXiv preprint arXiv:1909.06434*.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *ICML'15*, pages 2342–2350.

Michael Kampffmeyer, Arnt-Borre Salberg, and Robert Jenssen. 2016. Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1–9.

Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. 2015. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*.

Alex Kendall and Roberto Cipolla. 2016. Modelling uncertainty in deep learning for camera relocalization. In *2016 IEEE international conference on Robotics and Automation (ICRA)*, pages 4762–4769. IEEE.

Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR'18*, pages 7482–7491.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP'14*, pages 1746–1751.

Eliyahu Kiperwasser and Miguel Ballesteros. 2018. Scheduled multi-task learning: From syntax to translation. *Transactions of the Association for Computational Linguistics*, 6:225–240.

Iasonas Kokkinos. 2017. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6129–6138.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI'15*.

Changsheng Li, Junchi Yan, Fan Wei, Weishan Dong, Qingshan Liu, and Hongyuan Zha. 2017. Self-paced multi-task learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.

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

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017a. Adversarial multi-task learning for text classification. In *ACL'17*, pages 1–10.

Sulin Liu, Sinno Jialin Pan, and Qirong Ho. 2017b. Distributed multi-task relationship learning. In *ACM SIGKDD'17*, pages 937–946. ACM.

Zelun Luo, Yuliang Zou, Judy Hoffman, and Li F Fei-Fei. 2017. Label efficient learning of transferable representations acrosss domains and tasks. In *Advances in Neural Information Processing Systems*, pages 165–177.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. *arXiv preprint arXiv:1603.01354*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank.

Elliot Meyerson and Risto Miikkulainen. 2018. Pseudo-task augmentation: From deep multitask learning to intratask sharing—and back. *ICML'18*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Hyeonseob Nam and Bohyung Han. 2016. Learning multi-domain convolutional neural networks for visual tracking. In *ICPR'16*, pages 4293–4302.

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

Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. 2007. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP'14*, pages 1532–1543.

Anastasia Pentina, Viktoriia Sharmanska, and Christoph H Lampert. 2015. Curriculum learning of multiple tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5492–5500.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *ACL'17*, pages 2121–2130.

Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. 2017. Routing networks: Adaptive selection of non-linear functions for multi-task learning. *arXiv preprint arXiv:1711.01239*.

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

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017. Sluice networks: Learning what to share between loosely related tasks. *stat*, 1050:23.

Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2019. A hierarchical multi-task approach for learning embeddings from semantic tasks. In *AAAI'19*, volume 33, pages 6949–6956.

Tobias Scheffer, Christian Decomain, and Stefan Wrobel. 2001. Active hidden markov models for information extraction. In *Advances in Intelligent Data Analysis*, pages 309–318, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jürgen Schmidhuber. 1991. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227.

Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. 2013. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *EMNLP'08*, pages 1070–1079. Association for Computational Linguistics.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL'16*, pages 231–235.

Trevor Standley, Amir R Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. 2019. Which tasks should be learned together in multi-task learning? *arXiv preprint arXiv:1905.07553*.

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

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. 2015. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4068–4076.

Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *EMNLP'16*, pages 236–246.

Amir R Zamir, Alexander Sax, Nikhil Cheerla, Rohan Suri, Zhangjie Cao, Jitendra Malik, and Leonidas J Guibas. 2020. Robust learning through cross-task consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11197–11206.

Amir R. Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. 2018. Taskonomy: Disentangling task transfer learning. In *CVPR'18*, pages 3712–3722.

Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS'15*, pages 649–657.

Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.