Titre: Title:	Large Scale Conditional Multitask Learning for Natural Language Processing
Auteur: Author:	Amine El Hattami
Date:	2021
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	El Hattami, A. (2021). Large Scale Conditional Multitask Learning for Natural Language Processing [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/5613/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Large Scale Conditional Multitask Learning for Natural Language Processing

AMINE EL HATTAMI
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Février 2021

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Large Scale Conditional Multitask Learning for Natural Language Processing

présenté par **Amine EL HATTAMI**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Sarath CHANDAR ANBIL PARTHIPAN, président
Christopher J.PAL, membre et directeur de recherche
Laurent CHARLIN, membre externe

DEDICATION

To my family and friends,
For their encouragement and support

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Professor Christopher Pal for providing me guidance and support during my time at Polytechnique Montréal. Also, I want to thank the Montréal Institute of Learning Algorithms (MILA) team for providing me the platform (compute resources, office space, event organization, etc.) to better conduct my research in one of the best academic environments.

Thank you to Jonathan Pilault, PhD student at MILA for the amazing collaboration while working on the article. Finally, I would like to thank my family, closest friends and colleagues for supporting me throughout my time at Polytechnique Montréal.

RÉSUMÉ

Récemment, les “Masked Language Models” (MLMs) entrainés sur une large base de données non étiquetées ont généré des résultats de pointe pour plusieurs tâches dans le domaine du Traitement de Langage Naturel (NLP). Un grand nombre des soumissions les mieux classées au benchmark “General Language Understanding Evaluation” (GLUE) utilisent un modèle basé sur l’architecture “Bidirectional Encoder Representations from Transformers” (BERT). La formule “BERT + “fine tuning” par tâche individuelle” demeure prévalente dans la recherche ; cet outil permet en effet à plusieurs chercheurs d’aboutir à des résultats de pointe. Cependant, la méthode du “fine-tuning” manque d’efficacité quant à la quantité de paramètres utilisés, puisqu’il est probable qu’elle requière un nouveau modèle pour chaque tâche. De plus, cette méthode est susceptible d’entraîner la perte des connaissances acquises durant l’étape riche en données du pré-entraînement, ce qui pourrait affecter la performance de généralisation du modèle. L’Apprentissage Multi-Tâche (MTL) est une approche efficace par transfert inductif inspirée par les modes d’apprentissage de l’humain qui s’avère prometteuse dans le domaine du Traitement de Langage Naturel. Pourtant, le MTL n’atteint pas le même niveau de performance que l’approche du “fine-tuning” qui est encore appliquée après l’étape de l’Apprentissage Multi-Tâche dans plusieurs études. Ce manque de performance est dû aux difficultés additionnelles telles que le transfert négatif, le sur-apprentissage de tâches pauvres en données et la perte de connaissances.

Cette thèse a pour objectif d’explorer la possibilité de surpasser la performance de la méthode de “fine-tuning” en utilisant exclusivement l’Apprentissage Multi-Tâche. Cette recherche prendra en compte les défis d’optimisation connus de l’Apprentissage Multi-Tâche et des problèmes communs relatifs aux données. De plus, elle doit limiter du mieux possible la perte de connaissances acquises à l’étape du pré-entrainement, doit s’adapter facilement aux nouvelles tâches et être compatible avec tous les types de “Transformer”. Nous partons sur l’hypothèse qu’une architecture conditionnée par tâche est capable de gérer une configuration de MTL à grande échelle afin de maximiser le transfert entre toutes les tâches. L’utilisation d’une méthode d’échantillonnage basée sur la difficulté de chaque tâche pourrait gérer les différents niveaux de difficulté des tâches, ainsi que minimiser l’interférence et la perte de connaissances.

Nous proposons un modèle innovateur basé sur l’architecture “Transformer” que nous nommons "Conditional Adaptive Multitask Learning" (CA-MTL). Cette approche consiste en un nouveau mécanisme d’attention conditionnelle et un ensemble de modules conditionnés par

tâche facilitant le partage de paramètres. Par le biais de ce modèle nous atteignons un partage de paramètres plus efficace et limitons la perte de connaissances en fixant les paramètres du modèle pré-entraîné.

En utilisant CA-MTL, nous obtenons une meilleure performance que la méthode du “fine tuning” par tâche individuelle, tout en optimisant les données et les paramètres. Notre petit modèle a de ce fait obtenu une performance 2.2% supérieure à celle d’un grand modèle BERT pour le benchmark GLUE ; ces résultats ont été atteints en utilisant uniquement 64.6% des données et en ajoutant 5.6% de paramètres entraînables additionnels par tâche. Il est à noter ici qu’une méthode de “fine tuning” naïve peut ajouter jusqu’à 100% de paramètres entraînables. Notre approche s’est par ailleurs démontrée concurrentielle pour 26 tâches de Traitement de Langage Naturel, parvenant à des résultats novateurs sur plusieurs ensembles de test et de développement.

ABSTRACT

Recently, deep contextualized Masked Language Models (MLMs) trained on massive amount of unlabeled data pushed state-of-the-art (SOTA) results in many Natural Language Processing (NLP) tasks. Many leading submissions on the well-known General Language Understanding Evaluation (GLUE) benchmark are based on the Bidirectional Encoder Representations from Transformers (BERT) architecture. The formula "BERT + single task fine-tuning" continues to stay popular in many recent studies, constantly pushing SOTA results. However, fine-tuning is parameter inefficient since it may require a new model for each task. Moreover, it might *overwrite* the knowledge acquired during the data extensive pretraining, potentially hurting the generalization performance. Inspired by human ability to apply knowledge across tasks, multitask learning (MTL) is a powerful inductive transfer learning approach that showed promising in many NLP tasks. However, MTL alone doesn't reach the performance of fine-tuning approach and many studies still apply fine-tuning after the MTL step. This lack of performance is attributed to the additional challenges like negative transfer, overfitting of low resource tasks and catastrophic forgetting.

The goal of this thesis is to study the possibility of using MTL alone to outperform single task fine-tuning. This approach should address known MTL optimization challenges and data-related issues. It should limit pretraining knowledge loss and adapt to new tasks. Furthermore, the approach should be compatible with any Transformer architecture. We hypothesize that a task-conditioned pretrained architecture can handle a large-scale MTL setup, increasing the inductive transfer between tasks and leading to a better performance. Also, using a method to schedule task sampling based on its difficulty should help with varying task difficulties, task interference and avoid catastrophic forgetting.

We propose a novel Transformer based architecture named Conditional Adaptive Multitask Learning (CA-MTL) consisting of a new conditional attention mechanism and a set of task-conditioned modules facilitating weight sharing. Through this construction, we achieve more efficient parameter sharing and mitigate forgetting by keeping half of the weights of a pretrained model fixed. Using CA-MTL, we outperform single task fine-tuning while being, data and parameter efficient. Using only 64.6% of the data and adding only 5.6% more trainable parameters per task (a naive fine-tuning can add 100% of trainable parameters), our base model achieved 2.2% higher performance compared to a fine-tuned BERT large model on the GLUE benchmark. Our approach showed competitive on 26 NLP tasks achieving SOTA results on many test and development sets.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

BERT	Bidirectional Encoder Representations from Transformers
bi-LSTM	bidirectional Long Short-Term Memory
CA-MTL	Conditional Adaptive Multitask Learning
CRF	Conditional Random Fields
EGL	Expected Gradient Lenght
EM	expectation maximization
ER	Entity Recognition
GLUE	General Language Understanding Evaluation
GPT	Generative Pretrained Transformer
ICLR	International Conference on Learning Representations
LM	language model
ML	machine learning
MLM	Masked Language Model
MLN	Markov logic network
MT-DNN	Multi-Task Deep Neural Networks
MTAL	multitask active learning
MTL	multitask learning
NER	Named Entity Recognition
NLI	natural language inference
NLP	Natural Language Processing
PAL	Projected Attention Layer
POS	part-of-speech tagging
QBC	query-by-committee
SAN	Stochastic Answer Network
SNLI	Stanford Natural Language Inference
SOTA	state-of-the-art
SRL	Semantic Role Labeling
STL	single task learning
VSM	vector space model

CHAPTER 1 INTRODUCTION

1.1 Motivation

The use of pretrained contextualized language models (LMs) has shown a substantial performance improvement in various NLP tasks. Both in sentence-level tasks such as text classification [40], question answering [41] and natural language inference (NLI) [42], and token-level tasks such as Named Entity Recognition (NER), chunking [43], Semantic Role Labeling (SRL) [41] and constituency parsing [44].

With the introduction of the Transformer architecture [45], large-scale MLMs can learn general-purpose language representations in an unsupervised learning setup using the massive amount of unlabeled text data in multiple languages publicly available on the web. These models pushed SOTA results on multiple downstream NLP tasks [1, 46, 47] using fine-tuning. Specifically, these advances can be credited to the well known BERT [1] and its variants [48–50]. Recently, the formula "Bert-variant + single task fine-tuning" became popular in many studies, constantly pushing previous SOTA results on the GLUE benchmark[1]. Proving that fine-tuning large-scale MLMs is a powerful inductive transfer technique for NLP. The fine-tuning process might imply, creating a task-specific model by adding a few task-specific parameters generally in the form of a decoder head on the top of a BERT encoder [1]. Or, adding and training task-specific model components [37].

Fine-tuning can be parameter inefficient when training multiple downstream tasks. Since each task requires training an entire new model. Moreover, a recent study [51] showed that fine-tuning causes catastrophic forgetting of the knowledge acquired during the MLM pretraining phase. Thus, conserving this knowledge can improve the generalization performance of these models. Therefore, it's important to understand how this knowledge is stored. Multiple studies focused on analyzing the features that these models learn and their internal representations. [52] explored BERT's attention maps and showed that attention heads specialize in precise aspects of syntax and attributed BERT's success to its syntax aware attention. Another study [53] examined the captured linguistic information in a BERT model and discovered that its layers follow the traditional NLP pipeline, such that the syntactic information appears earlier in the network, while the semantic information appears at higher layers. The study also shows that the model can adjust this pipeline dynamically to enhance higher-level representations by updating lower-level decisions. [54] showed that the attention

[1] `https://gluebenchmark.com`

matrices encapsulate the grammatical representations and that the semantic information is encoded in low-dimensional subspace.

Motivated by humans' ability to use knowledge acquired from learning one task and apply it to a new task, MTL shows that the knowledge learned from each task can improve the performance on other *related* tasks. MTL has shown successful in multiple applications like NLP [55, 56], computer vision [57], speech recognition [58] and drug discovery [59]. MTL is an effective form of inductive transfer. It aims to improve the generalization of the representations by using domain-specific signals from *related* tasks trained in parallel while using a shared representation [60]. Inductive transfer uses domain-specific inductive bias introduced by the auxiliary tasks, which forces the model to select hypotheses that explain multiple tasks [33], that results in a better generalization. Moreover, adding new learning signals that introduce inductive biases can be less expensive than finding biases extracted from humans expertise [61].

However, MTL adds many optimization challenges such as different types of loss functions making it harder to achieve single task learning (STL) performance. MTL is also a balancing act when the dataset sizes and task difficulties differ. Too much sharing leads to negative transfer or destructive interference. While too little sharing, limits the possible cross knowledge.

1.2 Research Objectives

The objective of this thesis is to study the possibility of using MTL alone to outperform STL fine-tuning on various NLP tasks. Towards this end, we seek to build an approach that:

1. **Improves pretraining knowledge retention:** The approach should be able to retain as much pretraining knowledge as possible to retain and improve the generalization performance.
2. **Limits negative transfer and task interference:** The approach should limit negative transfer and task interference while maximizing the multitask inductive transfer. Also, it needs to deal with task difficulties imbalances.
3. **Is parameter efficient:** The approach should promote parameter sharing across all tasks while ensuring that adding a new task is dynamic (doesn't require an architecture change).
4. **Is data efficient:** The approach should use as much less data as possible to reduce the storage capacity and training time. It should also mitigate the data imbalance effect across tasks to avoid overfitting of low resource tasks.
5. **Adapts to new tasks:** The model should adapt well to new unseen tasks, since domain adaptation is a good indicator of the generalization performance.
6. **Adapts to new model architectures:** The approach shouldn't be tied to a specific model architecture.

1.3 Contribution

The contribution of this thesis consists of proposing and exploring the novel conditionally adaptive multi-task learning (CA-MTL) approach shown in Figure 3.1. To the best of our knowledge, our work is the first to:

1. To condition a pretrained MLM using a latent representation of tasks.
2. Apply uncertainty sampling for large scale MTL in NLP

Our CA-MTL approach consists of:

1. A novel multitask Transformer Attention Module using block-diagonal Conditional Attention (Section 3.3.1) that allows the Transformer original query-key attention to account for task-specific biases.

2. A novel set of conditional modules to adapt a pretrained MLM to new tasks:

    - A Conditional Alignment method that aligns the data of diverse tasks (Section 3.3.1).
    - A Conditional Layer Normalization module that adapts layer normalization statistics to each task (Section 3.3.1).
    - A Conditional Adapter that promotes weight sharing and task-specific signal flow from lower layers (Section 3.3.1).

3. A novel multitask sampling method to prioritize task selection using uncertainty to avoid uncertainty (Section 3.3.2).

We measure the efficacy of CA-MTL using 26 NLP tasks from various domains. We show that our approach achieves SOTA on multiple test and development sets.

This research was in collaboration with another researcher. However, with the help of my research director I initiated this project well before he joined. In this initial phase, I did an extensive literature review for the NLP tasks that we could use (60 tasks in total). I implemented all the MTL experimentation setup, including all the pre-processing code required. I also, ran multiple experiments that were used as a baseline for the research article.

The key idea behind this collaboration, was to join forces to produce best-in-class research, as illustrated by the acceptance of the paper on this research at the **International Conference on Learning Representations (ICLR) 2021** [62]. Also, it is worth mentioning that the order of first joint authors was randomly selected (coin flip) in the camera-ready version of the article.

During the collaboration, I was in charge of implementing the novel MTL sampling strategy and the novel conditional attention module. I contributed equally to the conditional layer normalization and participated in the discussions and refinement of all the other modules. On the experimentation side, I was in charge for running and analyzing the experimentation results of the MTL sampling strategy, the domain adaptation and the large-scale MTL experiment that achieved SOTA results. Moreover, I was in charge of the camera-ready version of our source code[2].

[2] `https://github.com/CAMTL/CA-MTL`

1.4 Thesis Outline

In Chapter 2, we provide relevant literature to understand the content of this thesis. In this overview, we mainly focus on the application of the discussed ML techniques in the context of NLP. First, we review transfer learning and MTL. Then active learning. After that, the transformers architecture. Finally, we review adapter architectures applied to transformers.

In Chapter 3, we present **the camera-ready version of the research article accepted in ICLR 2021** [62]. In this chapter, we detail the methodology we followed to build our approach (CA-MTL). Moreover, we show the detailed experimentation we performed to prove the efficacy of our method, including an ablation study for each on the novel components. Finally, in the appendix from the ICLR Paper, we present more experimental details and all the datasets used.

In Chapter 4, we review the thesis objectives and evaluate the extent to which our results achieved them.

Chapter 5 contains our conclusion summarizing the research, the known limitations of our proposed approach and our outlook for future work.

CHAPTER 2 LITERATURE REVIEW

2.1 Transfer Learning

In a traditional supervised ML setup, it's assumed that labeled data from a task and domain A is available to train a model for the same task and domain. In this setup, a new task and domain B require that labeled data be available to train the *new* model from scratch as shown in Figure 2.1 (a). However, supervised learning falls short when labeled data for a new task is missing, scarce, easily outdated or expensive to gather. Inspired by human ability to transfer previously learned knowledge to solve new tasks, transfer learning is a ML technique that aims to use knowledge acquired from a task and domain defined respectively as the *source task* and *source domain* to help learn a new task and domain defined respectively as the *target task* and the *target domain*, as shown in Figure 2.1 (b).

2.1.1 Definition

We provide a definition of transfer learning following the work of [31]. Transfer learning relies on the concepts of a *domain* and a *task*. A domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$ consists of a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$ over the feature space, where $X = \{x_1, ..., x_n\} \in \mathcal{X}$ and x_i is the feature representation of the i-th sample. X is a random variable that represents the training samples.

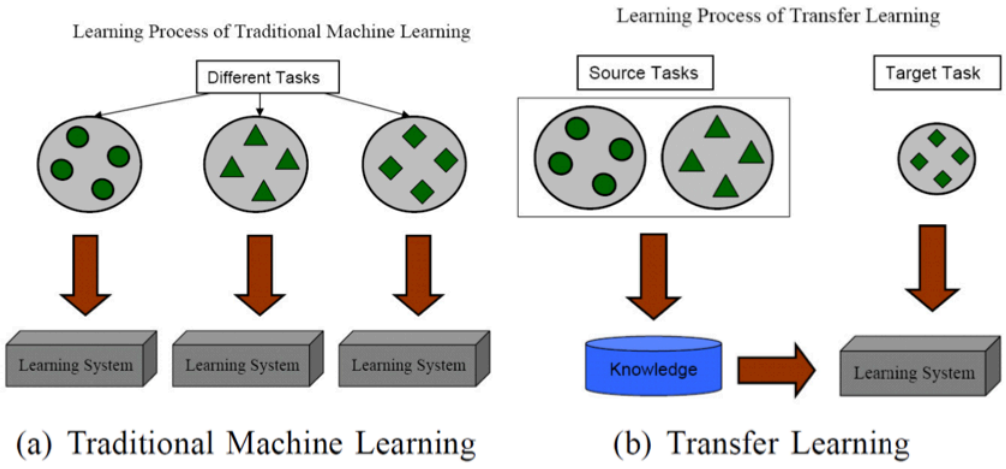
Given a domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$, a task $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$ consists of a label space $\mathcal{Y}$ and an objective predictive function $f(.)$. The function $f(.)$ represents the conditional probability distribution $P(Y|X)$. Once $f(.)$ is learned from a set of training samples pairs $\{x_i, y_i \ | x_i \in X, y_i \in \mathcal{Y}\}$, it can predict the label $f(x)$ of an unseen instance x.

Definition 1 *Transfer learning is a ML technique that aims to improve the learning of the target conditional probability distribution $P_T(Y_T|X_T)$ for a target task $\mathcal{T}_\mathcal{T} = \{\mathcal{Y}_\mathcal{T}, P(Y_T|X_T)\}$ and a target domain $\mathcal{D}_T = \{\mathcal{X}_T, P(X_T)\}$ using knowledge learned from a source task $\mathcal{T}_\mathcal{S} = \{\mathcal{Y}_\mathcal{S}, P(Y_S|X_S)\}$ and a source domain $\mathcal{D}_S = \{\mathcal{X}_S, P(X_S)\}$, where $D_T \neq D_S$, or $\mathcal{T}_T \neq \mathcal{T}_S$.*

In the above definition, $D_T \neq D_S$ implies that either $\mathcal{X}_\mathcal{T} \neq \mathcal{X}_\mathcal{S}$ or $P(X_T) \neq P(X_S)$ and $T_T \neq T_S$ implies that either $\mathcal{Y}_T \neq \mathcal{Y}_S$ or $P(Y_T|X_T) \neq P(Y_S|X_S)$. In the following section, we describe each scenario in more details.

1. $\mathcal{X}_\mathcal{T} \neq \mathcal{X}_\mathcal{S}$: The feature space between the target domain $\mathcal{D}_T$ and source domain $\mathcal{D}_S$ are different.

Figure 2.1 Difference between traditional ML and transfer learning as presented in [31]



(a) Traditional Machine Learning (b) Transfer Learning

2. $P(X_T) \neq P(X_S)$: The marginal probability distributions of the target domain $\mathcal{D}_T$ and source domain $\mathcal{D}_S$ are different, where $X_{S_i} \in \mathcal{X}_S$ and $X_{T_i} \in \mathcal{X}_T$, but the feature spaces are the same. This scenario corresponds to *domain adaptation* [63].

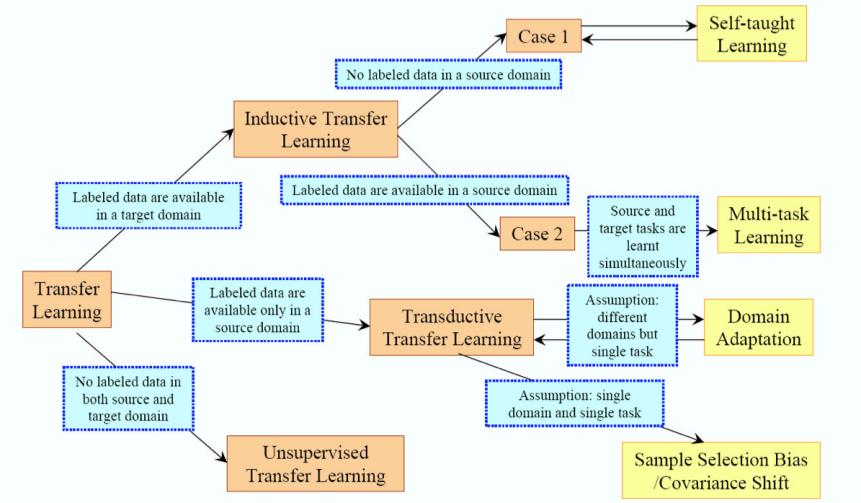
3. $\mathcal{Y}_T \neq \mathcal{Y}_S$: The label space between the target task $\mathcal{T}_T$ and source task $\mathcal{T}_T$ are different. In this case we can make two distinctions. Either *multitask learning* [60], where we learn tasks simultaneously, or sequentially as described in [6,64], where the source task is also referred to as an *intermediate* task.

4. $P(Y_T|X_T) \neq P(Y_S|X_S)$: The conditional probability distributions between the target task $\mathcal{T}_T$ and source task $\mathcal{T}_T$ are different, where $Y_{S_i} \in \mathcal{Y}_S$ and $T_{T_i} \in \mathcal{Y}_T$.

2.1.2 Taxonomy

Based on the above definition and the different scenarios between the target and source tasks and domains, transfer learning consists of three categories, as shown in Figure 2.2. (1) *Inductive transfer learning* ($T_T \neq T_S$) is where the target and source tasks are different, regardless the relationship between the target and source domains. (2) *Transductive transfer learning*, where the target and source domains are different but the target and source tasks are the same ($\mathcal{D}_T \neq \mathcal{D}_S$ and $T_T = T_S$). (3) *Unsupervised transfer learning* where the target and source task are different but *related*. The following section describes each category in more details.

Figure 2.2 Different transfer learning scenarios as presented in [31]



Inductive Transfer Learning

In this setting, *inducing* the target objective predictive function requires some target domain labeled data. The case where labeled source domain data is available is like a multitask learning setup. However, a multitask learning setup aims to achieve a good performance on all tasks learned simultaneously, while the inductive transfer learning aims to improve the learning of the target task regardless of the performance of the source task. The case where only unlabeled data is available in the source domain is like *self-taught learning* [65]. In a self-taught learning setup, the target and source data aren't assumed to have the same class labels or label distribution, making use of data from any source to improve the performance on the target task.

Definition 2 *Inductive transfer learning is a ML technique that aims to improve the learning of the target conditional probability distribution* $P_T(Y_T|X_T)$ *in the target domain* $\mathcal{D}_T$ *using knowledge learned from a source task* $\mathcal{T}_S$ *and domain* $\mathcal{D}_S$*, where* $\mathcal{T}_T \neq \mathcal{T}_S$*.*

Transductive Transfer Learning

In this setting, a lot of labeled source domain data is available while labeled target domain data isn't available. Depending on the differences between the target and source domains, the transductive transfer learning consists of two sub-cases. (1) Case where the feature spaces between the target and source domains are different ($\mathcal{X}_\mathcal{T} \neq \mathcal{X}_\mathcal{S}$) . (2) Case where the marginal probability distribution of the input data is different, but the feature spaces between

the target and source domains are the same ($P(X_T) \neq P(X_S)$ and $\mathcal{X}_\mathcal{T} = \mathcal{X}_\mathcal{S}$). Example, for this case are sample selection bias [66] or co-variate shift [67] and *domain adaptation* for text classification tasks [68].

In the ML literature, the term *transductive* has several definitions. [69] Introduced the term transductive transfer learning, where it required that the target task and the source task are the same regardless of the domains. Also, requires all the test data at training time. But the definition provided in [31] only requires part of the unlabeled data at training time with the assumption that is enough to estimate the marginal probability for the target data.

Definition 3 *Transductive transfer learning is a ML technique that aims to improve the learning of the target conditional probability distribution $P_T(Y_T|X_T)$ in the target domain $\mathcal{D}_T$ using knowledge learned from a source task $\mathcal{T}_\mathcal{S}$ and domain $\mathcal{D}_S$, where $\mathcal{D}_T \neq \mathcal{D}_S$ and $T_T = T_S$. Moreover, it requires some unlabeled target domain data during training.*

Unsupervised Transfer Learning

In the unsupervised transfer learning, only unlabeled data is available for training for both the target and source domains. The unsupervised transfer learning focuses on unsupervised tasks like density estimation , clustering [65] and dimensionality reduction [70].

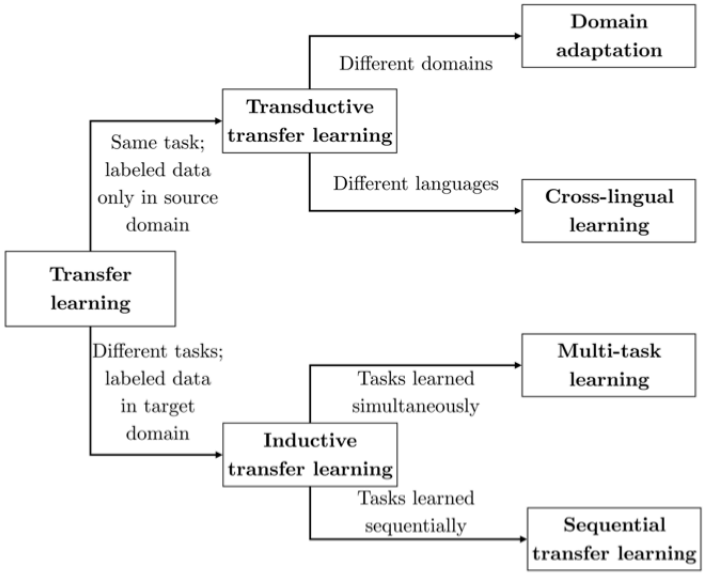
Definition 4 *Unsupervised transfer learning is a ML technique that aims to improve the learning of the target conditional probability distribution $P_T(Y_T|X_T)$ in the target domain $\mathcal{D}_T$ using knowledge learned from a source task $\mathcal{T}_\mathcal{S}$ and domain $\mathcal{D}_S$, where $T_T = T_S$ and $\mathcal{Y}_T$ $\mathcal{Y}_S$ aren't observable.*

Taxonomy for Natural Language Processing

In Figure 2.3 we show a summarized transfer learning taxonomy applied to NLP as presented in [32]. This summary is based on the one presented in [31] with the following adjustments:

- In the context of NLP, *cross-lingual learning* (or *cross-lingual adaptation*) [71] is when the feature space between the target and source domains are different. In this case the samples from the domains are in two languages.
- The sample selection bias and co-variate shift fall in the domain adaptation case.
- To make a clear separation from MTL, *sequential transfer learning* includes the self-taught and unsupervised transfer learning.

Figure 2.3 Different transfer learning scenarios as presented in [32]



2.1.3 Transfer Techniques

The categories above describes the techniques on "How to transfer." This section we will go over the techniques to transfer knowledge across tasks and domains, answering the question "What to transfer" as described in [31].

Instance Transfer

Instance transfer (or instance-based transfer) technique uses some source domain data to aid learning the target task. Instance transfer relies on two main approaches (1) Instance re-weighting and (2) importance sampling.

Instance re-weighting is a mechanism alters the weights of training samples to adjust their impact, used when labeled data from a different but related source domain is available while labeled data from the target domain is scarce. An example of this technique is the Transfer AdaBoost (TrAdaBoost) [72] which is an application of AdaBoost [73] to transfer learning. TrAdaBoost "aims to boost the accuracy of a weak learning by carefully adjusting the weights of training instances and learn a classifier accordingly" [72]. This framework divides the training data into two sets. (1) *Same-distribution* training data that have the same distribution as the test data and is assumed not enough to train the model and (2) *diff-distribution* training data that have a different distribution than the test data and is

assumed to be abundantly available. The framework uses AdaBoost for the same-distribution data. However, when the diff-distribution training data is wrongly predicted, TrAdaBoost has a mechanism that "decreases the weights of these instances in order to weaken their impacts" [72].

Importance sampling is an approximation method used instead of sampling. Equation 2.1 describes the expectation of a target objective predictive function $f_T(x)$, where x follows the distribution $p_T(x)$.

$$E[f_T(x)] = \int f_T(x)p_T(x)dx \approx \frac{1}{n}\sum_i f_T(x_i) \tag{2.1}$$

In the case where, enough labeled data from the target domain is available, the Monte Carlo sampling method [74] to calculate the expectation of the objective predictive function $f_T(x)$ by sampling x from $p_T(x)$. However, when target domain labeled data is scarce or hard to sample from, importance sampling can estimate the expectation using the source domain distribution $p_S(x)$ where enough labeled data is available as shown in equation 2.2, where $p_S(x)$ is the source domain distribution.

$$E[f_T(x)] = \int f_T(x)p_T(x)dx = \int f_T(x)\frac{p_T(x)}{p_S(x)}p_S(x)dx \approx \frac{1}{n}\sum_i f_T(x_i)\frac{p_T(x_i)}{p_S(x_i)} \tag{2.2}$$

In equation 2.2, $\frac{p_T(x)}{p_S(x)}$ is known as the sampling ratio (or sampling weight) and acts as a correction ratio that corrects the offset introduced by sampling from a different distribution. The importance sampling estimation method can suffer from a large variance when the target and source domain are very different. Equation 2.3 is the variance of estimation, where $X = f_T(x)\frac{p_T(x)}{p_S(x)}$. Thus, a large $\frac{p_T(x)}{p_S(x)}$ leads to a large variance.

$$Var(X) = E[X^2] - E[X]^2 \tag{2.3}$$

Feature Representation Transfer

The feature representation is a knowledge transfer method that maps the feature representation from the source domain to the target domain [75]. The main difference between other transfer methods is that it uses both the feature-based method and the case-based method to achieve the knowledge transfer for text classification tasks. The method has three steps. (1) Using the feature-based method, create a feature representation subspace common to both the target and source domains. The idea is to find the shared components between the domains that doesn't affect the distribution across domains and "capture the structure of the original data well" [75]. Example, two text classification datasets can have different domains.

However they share many features (words). (2) Using the case-based method, learn a feature representation mapping function in the new feature subspace. To minimize the distance between the two distributions, the source and domain data are re-weighted. After that, create a vector space model (VSM) using the mapping function to fit the distribution in the target domain. (3) Use any traditional ML algorithm to train a text classifier in the source domain and apply it for the target domain.

Parameter Transfer

In the parameter transfer approach, it's assumed that a parametric model exists in both the target and source domains and that, parameters embed the transferred knowledge. These parameters are estimates using samples from the source domain. However, techniques like self-taught learning [65] is applied when labeled data isn't available. These techniques yield good performance when unlabeled data is abundantly available. Recently, fine-tuning deep neural networks trained on source domains started using the parameter transfer approach. The approach can also be used to multiple kernel learning [76] and sparse coding [77].

Relational Knowledge Transfer

The relational knowledge transfer [78] is a technique used for relational domains. The main assumption is that there exists a relationship between the target and source domains. This approach uses Markov logic network (MLN) for learning relational domains. The transfer technique has two steps. (1) *predicate mapping*: create the best mapping between the target domain data and the source domain data. Example, a predicate mapping between a book dataset and a research paper dataset might learn that book authors are like students and that books are like research papers. There exist two general approaches to create this mapping. (a) A *global mapping*, where the entire source MLN is translated using a mapping created for each source predicate to a target predicate. (b) A *local mapping* that finds the best mapping for each source clause by creating a mapping for the predicated in each clause. The relational knowledge transfer uses the local mapping method since the global mapping method isn't easily scalable for large numbers of predicates. Using the mapping, clauses from the source domain are translated to the target domain. However, the translation can introduce erroneous clauses, thus requiring a refinement. (2) *theory refinement*: The newly translated clauses are revisited and re-weighted to properly model the target domain data. This step follows the theory refinement method [79], except that the framework learns the revised theory instead of receiving it from a human expert.

2.1.4 Negative Transfer Learning

As seen in the previous sections, transfer learning is a powerful ML technique that has shown effective, especially when labeled data is scarce for a desired target task. Negative transfer is the phenomenon that arises when applying transfer learning hurts the performance of the target task. Multiple works studied how to avoid and detect this phenomenon without any external input. Using a "transfer-aware" naive Bayes classification algorithm, [80] conducted an empirical study to show the existence of negative transfer when the target domain isn't sufficiently related to the target domain. Formally, the lack of "relatedness" happens when the joint distribution of the source domain $P_S(X, Y)$ diverges from the target domains distribution $P_T(X, Y)$, where X is the input random variable and Y is output [81]. The situation when the marginal probability of the source and target domain are equal ($P_S(X) = P_T(X)$) and $P_S(Y|x)$ is uniform for any x, represents a case where no meaningful knowledge can be extracted from $P_S(X, Y)$. In this case applying transfer learning will surely introduce negative transfer unless $P_T(Y|X)$ it's also uniform. [82] offered a formal framework for measuring task relatedness. This framework derives a strong generalization bounds and provides the general conditions that guarantees the generalization compared to the bounds of single task learning method. In [83], the tasks are grouped by either a task clustering or a task gating method. Using the task clustering method, it's assumed that there exists multiple clusters that group-related tasks instead of a single cluster. This method relies on estimating the posterior data likelihood to assign tasks to the most compatible cluster and use the expectation maximization (EM) algorithm to optimize the likelihood of the shared parameters. The task gating technique adds a task-dependent feature. This addition removes the disadvantage of using the task clustering technique where all tasks are assigned with the same probability to each cluster. Thus, making the assignment of a task to a cluster task-dependent. The task gating method represents a generalization of the task clustering in the case where all tasks share the same feature. Even if transfer learning is a widely used technique. It's still a hard to detect, understand or avoid negative transfer especially when the target domain data is tiny.

2.2 Multitask Learning

Standard ML setup consists of learning a new model for each single task of interest, referred to as STL. In a STL setup, the optimization usually involves a single loss function. Moreover, large tasks are divided into multiple sub-tasks and are learned separately. For example, a chatbot system will have a separate model for each task (intent classification, NER, ...) trained separately. However, STL can benefit from the knowledge learned from other *related*

tasks. Thus, when training all tasks in parallel, each task should benefit from the knowledge of other related tasks. This setup is MTL [61].

In the ML literature, the term MTL is used for multiple settings. However, we don't consider the learning with auxiliary tasks setting nor the setting where the number of tasks changes during training. Instead, we focus our review to the setting in which a fixed set of tasks is learned jointly and all the tasks are treated equally. Moreover, MTL can be used with other learning approaches like semi-supervised learning, unsupervised learning, Active Learning (AL) and Reinforcement Learning (RL). However, we focus on the supervised learning setup, following most of MTL work.

MTL is also referred to as learning to learn and joint learning. MTL is a form of inductive transfer that aims to improve the generalization performance by using domain-specific knowledge extracted from the training signals of related tasks drawn from the same domain. Inductive transfer enhances the generalization performance and the learning speed. This form of transfer provides an inductive bias (training signals) extracted from other tasks. An inductive bias helps an inductive learner to favor hypotheses that explains multiple objectives. The training signals from other tasks represents a bias since "when the training signals are for tasks other than the main task, it is easy to see that, from the point of view of the main task, the other tasks may serve as a bias" [84]. MTL trains all the tasks in parallel that share a hidden layer. The shared representation helps learning features that describe multiple tasks at the same time. MTL is inspired by the human learning behavior where we (humans) often use previously acquired knowledge to enhance learning a novel task. For example, the case where a person is learning Portuguese and Spanish (two Latin languages) at the same time. Knowledge acquired while learning Portuguese can be used to enhance and accelerate learning Spanish and vice versa.

As seen in the previous sections, MTL is similar to transfer learning. However, there is a significant difference. Transfer learning aims to improve the performance of the target task regardless of the performance of the source task. The source task is seen as a *helper* task. In the other hand, MTL aims to improve the performance of all tasks jointly. However, [85] introduced the *asymmetric* MTL that aims to improve the performance of a target task using multiple source tasks that were trained jointly beforehand. The symmetric MTL is closer to transfer learning, however, the source tasks are still learned simultaneously.

2.2.1 Definition

We provide a formal definition of MTL following the work of [86].

Definition 5 *Given T learning tasks over an input space $\mathcal{X}$ and a set of task spaces $\{\mathcal{Y}^t\}_{t\in[T]}$, such that a training dataset of i.i.d data points $\mathcal{D}_i = \{x_j^t, y_j^t\}_{j=1}^{n_t}$ for each task $t \in [T]$, where x_j^t is the jth training sample, y_j^t is its label and n_t is the amount of training samples. Each task has an objective predictive function $f^t(x; \theta^{shared}, \theta^t) : \mathcal{X} \to \mathcal{Y}^t$, where θ^{shared} are the shared parameters between all tasks, while θ^t are task-specific parameters. And $\mathcal{L}^t(.,.) : \mathcal{Y}^t \times \mathcal{Y}^t \to \mathbf{R}^+$ is the task-specific loss function. MTL aims to improve learning each task using knowledge from all or some of the tasks with the objective in Equation 2.4, where c^t is a per-task weight either learned or provided as a hyper parameter and $\hat{\mathcal{L}}^t$ is the empirical loss of a task t as defined in Equation 2.5*

$$\min_{\theta^{shared},\theta^1,\ldots,\theta^T} \sum_{t=1}^{T} c^t \hat{\mathcal{L}}^t(\theta^{shared}, \theta^t) \tag{2.4}$$

$$\hat{\mathcal{L}}^t(\theta^{shared}, \theta^t) \triangleq \frac{1}{\sum_{t=1}^{T} n_t} \sum_i \mathcal{L}(f^t(x_i; \theta^{shared}, \theta^t), y_i^t) \tag{2.5}$$

2.2.2 Multitask Learning Techniques

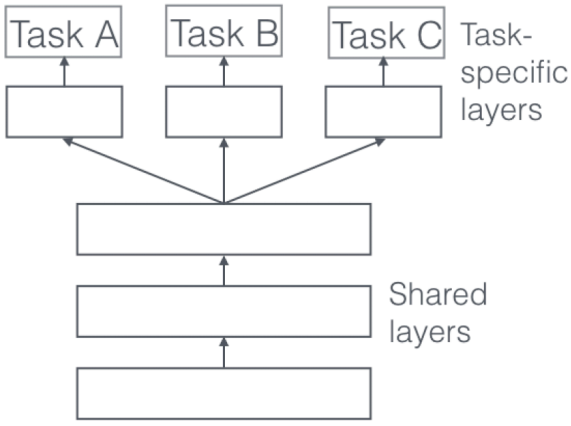
There are two main techniques used for MTL in the context of neural networks: *hard parameter sharing* or *soft parameter sharing*. In both cases, the model tries to lean a shared or a similar hidden representation for all tasks based on some constraint or relationship regularization between tasks. The following section will describe each one these techniques in more details.

Hard Parameter Sharing

Hard parameter sharing is the most widely used technique and the easiest to implement for MTL in the context of neural networks. First introduced in [61], each task has it own task specific output layers, however, all tasks share the same hidden layers, as shown in Figure 2.4. In this setup, the model tries to learn a shared feature presentation that models all the tasks. [87] showed that hard parameter sharing reduces the risk of overfitting by "an order T smaller than overfitting the task-specific output layers where T is the number of tasks" [32].

In the context of NLP, MT-DNN [4] achieved SOTA results (at the moment of its publication) on the GLUE benchmark [88], Stanford Natural Language Inference (SNLI) [89] and SciTail [90] by using the hard parameter sharing technique. MT-DNN architecture as shown in Figure 2.5 shares the lower layers between all tasks which consist of a BERT encoder and the task-specific top layers.

Figure 2.4 Hard parameter sharing for MTL in neural networks as presented in [33]



For single-sentence classification, The probability that X belongs to class c is predicted by a logistic regression with softmax (Equation 2.6), where W_{SST} is the task-specific matrix.

$$P(c|X) = \text{softmax}(W_{SST}^T \cdot \text{x}) \tag{2.6}$$

For text similarity tasks, The similarity of two sentences is calculated based on the semantic representation of the sentence pair (Equation 2.7) , where W_{STS} is the task-specific matrix and $\text{Sim}(X_1, X_2) \in \mathbf{R}$.

$$\text{Sim}(X_1, X_2) = W_{STS}^T \cdot \text{x} \tag{2.7}$$

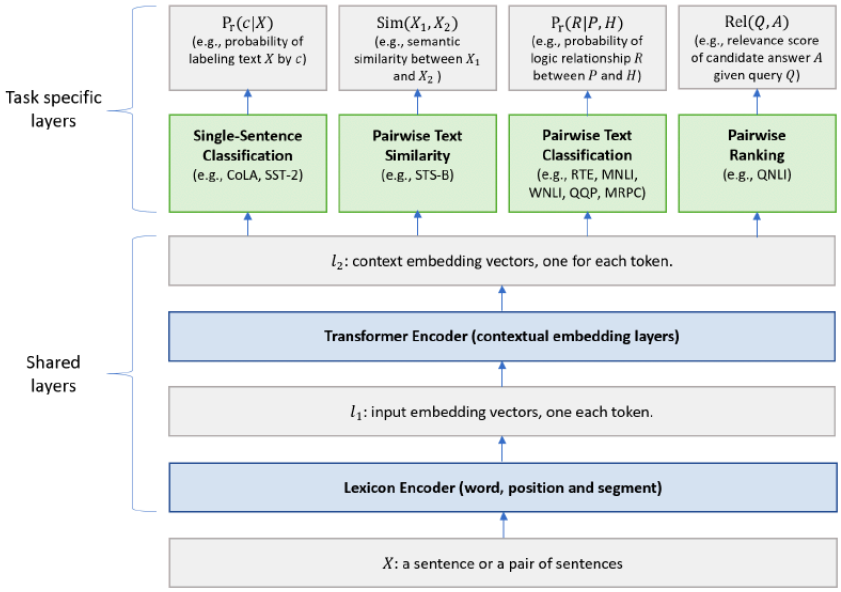
For relevance ranking tasks, The relevance is calculated using the semantic representation of the question Q and candidate answer A pair (Equation 2.8), where W_R is the task-specific matrix. For a given Q the answer with the best relevant score is selected.

$$\text{Rel}(Q, A) = g(W_R^T \cdot \text{x}) \tag{2.8}$$

For pairwise text classification tasks, Stochastic Answer Network (SAN) [91] is used to predict the relation between the premise and hypothesis.

[34] also used hard parameter sharing to jointly train a main semantic sequence labeling task with multiple auxiliary tasks. They used an off-the-shelf bi-LSTM model [92]. The model is shown in Figure 2.6. However, only the lower stacked bi-LSTM are shared between all tasks and output layers for each one of the tasks is added using a softmax.

Figure 2.5 MT-DNN architecture as presented in [4]



Soft Parameter Sharing

Soft parameter sharing implies that each task has its own model. In this setup, we learn a model of each task while aiming to reduce the distance between the parameters of each model using a regularization method, as shown in Figure 2.7. In this setup the shared feature space representation is loosely coupled.

As a realization mechanism, [35] augmented the learning objective to include a regularization coefficient to impose the similarities between the corresponding parameters in a cross-lingual parameter sharing setting, as shown in Figure 2.8 . The following is a simplified version where only two tasks (A and B) are in the MTL setup. Equation 2.9 represents the augmented loss function, where $\mathcal{L}^{'}$ is the original multitask loss function (e.g., the sum of losses), N is the total amount of training samples, $W_i^{(t)}$ denotes the ith layer parameter of tasks t, $\|\cdot\|_F^2$ is the squared Frobenius norm and λ is a sensitivity hyper-parameter.

$$\mathcal{L} = \mathcal{L}^{'} - \sum_{i=1}^{N} \lambda \|W_i^{(A)} - W_i^{(B)}\|_F^2 \tag{2.9}$$

[93] regularised the parameters for all models using the tensor trace norm, to promote parameter reuse between all models in a MTL setup. Initially, this method was used to replace rank constraints in Reduced Rank Regression [94]. The framework doesn't impose a

Figure 2.6 Multitask bi-LSTM architecture as presented in [34]. Solid lines represent the main task, while the dashed lines represent the auxiliary tasks (for clarity, only auxiliary task one is shown)

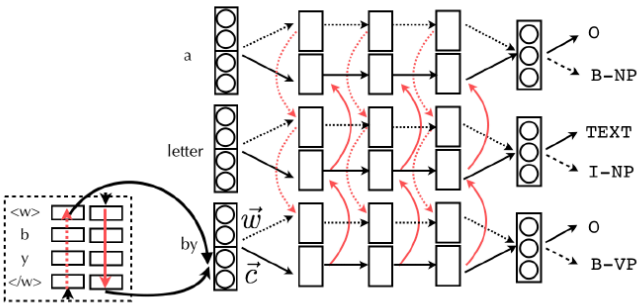
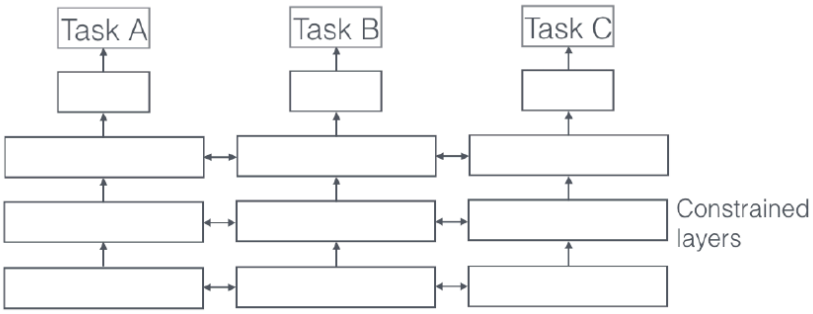


Figure 2.7 Soft parameter sharing for MTL in neural networks as presented in [33]

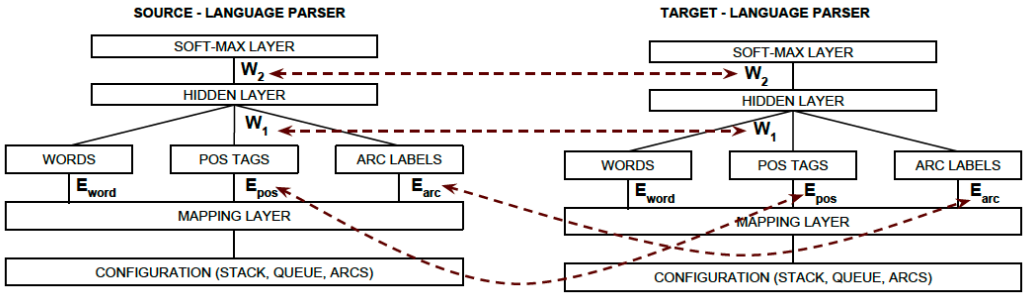


parameter sharing strategy (which layers should be tied), instead it uses a tensor norm on the parameters of each layer so that the sharing strategy is learned from the training data. The trace norm is the sum of singular values of a matrix $\|X\| = \sum_{i=1} \sigma_i$. Minimizing the trace norm (or nuclear norm) replaces the need to finding a minimum rank solution when it is hard to calculate [95]. [93] presented two methods to define the tensor rank. (1) Using the Tucker decomposition [96] as defined in Equation 2.10, where $\mathcal{W}_{(i)}$ is the mode-i tensor flattening [97] and (2) using the Tensor Train decomposition [98] as defined in Equation 2.11, where $\mathcal{W}_{[i]}$ is another flattening method.

$$\|\mathcal{W}\|_* = \sum_{i=1}^{N} \gamma_i \|\mathcal{W}_{(i)}\|_* \tag{2.10}$$

$$\|\mathcal{W}\|_* = \sum_{i=1}^{N-1} \gamma_i \|\mathcal{W}_{[i]}\|_* \tag{2.11}$$

Figure 2.8 Soft parameter sharing for language parser, as presented in [35]



2.2.3 Limitations of Multitask Learning

Compared to STL, MTL adds a number of optimization challenges, making it harder to achieve STL performance. MTL is a balancing act when dealing with the amount of knowledge to be shared between tasks. Too much sharing leads to negative transfer (or destructive interference) which hurts the performance of one or all tasks. While too little sharing limit the possible cross knowledge. A good MTL model is the one that finds that fine balance. Recent work studied how to detect and how to avoid negative transfer in a MTL setup. However little work has proposed in NLP. [99] showed that conflicting gradient between tasks is predictive of negative interference. Also, proposed a model-agnostic algorithm (PCGrad). In this algorithm, two gradients g_i and $g_j$and are said to be *conflicting* if the cosine similarity between the two gradients is negative. If the cosine similarity is positive, the gradients are left untouched. Else g_i is projected onto the normal plane of g_j.

2.3 Active Learning

The core idea behind Active Learning is that an algorithm can achieve better performance using less labeled data if it's able to selects its training data, thus reducing labeled data and compute requirements. AL (or query learning) is well suited when labeled data is hard or expensive to gather while unlabeled data is abundantly available (e.g., NLP where unlabeled text data is massively available[1]).

An active learner *queries* an *oracle* (e.g., a human annotator or automated process) to label some of the unlabeled data to help its learning. There are three major approaches to AL in the ML literature. (1) membership query learning, and (2) Pool-based sampling and (3) stream-based selective sampling.

[1]Recently a 10 TB text dataset was made public at https://github.com/EleutherAI/The-Pile

2.3.1 Membership Query Learning

Membership queries [100] is one of the first Active Learning techniques. The learner can *query* labels for any unlabeled data instance, even those that are already seen instead of sampling from the input underlying distribution. Once selected an oracle provides the label. It is easy to see that such technique is very demanding when the oracle is a human annotator. For fields like scientific discovery membership query , is a promising technique for fields like if the oracle is an automated process this. [101] used a "robot-scientist" as the oracle to perform the experiments queried by the model. This yield to a big decrease of experimental cost.

2.3.2 Pool-Based Sampling

In pool-based sampling [102], the training dataset is composed by an initial small set of labeled data $\mathcal{I}$ and a large set of unlabeled data $\mathcal{U}$. Queries are selected from the unlabeled instances using an informativeness metric. The metric helps selecting the unlabeled training samples *worth* labeling.

Most pool-based sampling variations fit a common algorithm shown in Algorithm 1. The informativeness and ranking function (h) is the main difference between pool-based sampling methods. Some variations do not require training the classifier on the initial labeled set. Also, the training ends either the labeled set reached a desired size or no unlabeled data is left. Moreover, adding one sample at each round it most widely used approach ($n = 1$).

Algorithm 1: General Pool-Based Active Learning Algorithm

Input: Initial training set $\mathcal{I}$, Unlabeled data set $\mathcal{U}$, Informativeness and ranking function h, labeled samples per round n

$\mathcal{D} \leftarrow \mathcal{I}$ // Initialize the training data with the initial labeled data;

Train the classifier on $\mathcal{D}$;

while *Train stopping requirement not met* **do**

  Select $\mathcal{S}$ random samples from $\mathcal{U}$;

  $\mathcal{S}_{n-\text{top}} = h(S, n)$ //Calculate informativeness value and return n-top ranked samples;

  Use oracle to label $\mathcal{S}_{n-\text{top}}$ samples;

  $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{S}_{n-\text{top}}$// Remove labeled samples from $\mathcal{U}$;

  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{S}_{n-\text{top}}$ // Append labeled samples to $\mathcal{D}$;

end

2.3.3 Stream-Based Selective Sampling

In Stream-bases selective sampling [103] the learner draw unlabeled samples from the actual underlying input space distribution, then decided whether to discard or ask for their labels. It is also called sequential AL since the instances are sampled individually. This technique is similar to membership query learning when the input space distribution is uniform.

There are several methods to decide whether to keep a drawn sample or not. A naive approach is to set a minimum threshold on an informativeness metric. Another way is to use an informativeness metric so that samples with higher informativeness value are more likely to be kept (biased decision) [104]. Another approach is to find the regions where the model is uncertain [105]. An unlabeled sample is from an uncertainty region if two different classifiers with the same model class agree on all the labeled samples but disagree on this sample. Multiple approximation approaches [106, 107] were introduced to calculate the uncertainty region since the verification between the two models needs to be recalculated after labeling each new sample which can be very expensive.

2.3.4 Query Strategies

A measure of informativeness (or query strategy) is required for all AL approaches described above. Multiple strategies were proposed in the ML literature like uncertainty sampling [102], query-by-committee (QBC) [108], Expected Gradient Lenght (EGL) [109], expected error reduction [110], variance reduction and information density [111]. However, we focus uncertainty sampling since its most related to our work.

Uncertainty Sampling

[102] introduced the term uncertainty sampling. Uncertainty sampling is a heuristic used with probabilistic classifiers to help selecting the least certain samples. It's similar to incremental training [112] where a classifier is retrained on misclassified samples. However, uncertainty sampling uses a classifier trained on an initial pool of labeled data to calculate the uncertainty of the unlabeled data with the intuition that a high uncertainty indicates a decision boundary, and learning a classifier aim to clarify the position of the decision boundary.

Uncertainty sampling increases the cost of sampling by $O(TD)$, where T is the number of candidates, D the number of predictors.

There are multiple ways to calculating uncertainty. We will present two variants: entropy

sampling and margin sampling.

Entropy sampling selects the samples with highest Shannon entropy [113] value following Equation 2.12, where q_c is the probability of an instance to belong to class c. Here, Algorithm 1 would rank the samples with the highest values on top.

$$H(X) = -\sum_{c} q_c \log q_c \tag{2.12}$$

For a binary classification, entropy sampling aims to selects the samples with highest entropy. Specifically, instances with class posterior closest to 0.5, where entropy is at its highest value as shown in Figure 2.9. Entropy sampling is easily applied to probabilistic multi-class classifiers, sequences [111] and trees [114].

Margin sampling [115] selects the samples with the lowest margin between the most probable classes as shown in Equation 2.13, where c_1 and c_2 are the most probable classes for the sample x_i. Intuitively, the margin value is small when the classifier is able to make a clear separation between the classes. In contrast, when the margin value is large the classifier would benefit from knowing the true class in order to better discriminate between these classes. However, this method ignores the remaining classes (classes not used to calculate the margin value) when the number of classes is big. Here, 1 would rank the samples with the lowest values on top.

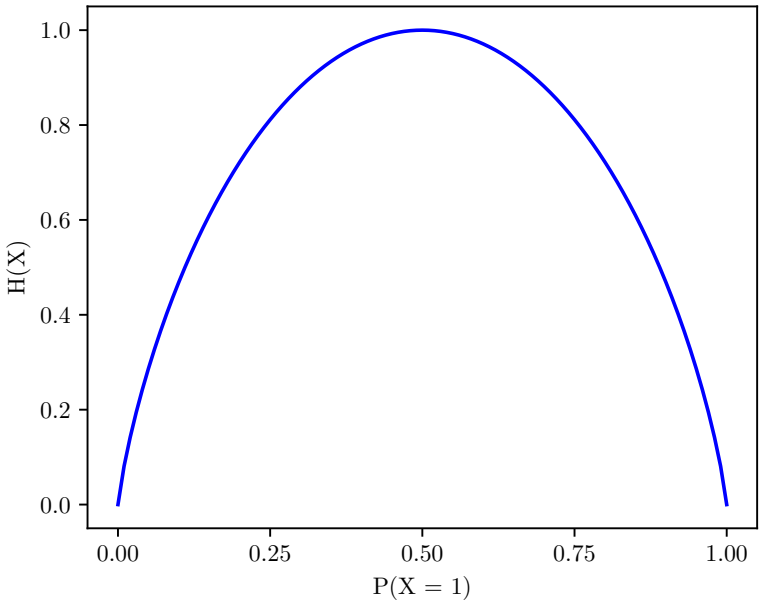
$$M_i = |P(c_1|x_i) - P(c_2|x_i)| \tag{2.13}$$

Both entropy and margin sampling select samples where the posterior distribution is most uniform. However, they differ in the probability space. Entropy sampling doesn't favor samples where one of the labels is very unlikely, since the classifier is certain that it's not the true class. Instead, it considered useful when using margin sampling when the classifier is not able to differentiate between the remaining classes. Selecting the right sampling method is rather application dependent [111, 116]. However, it is easy to say that it is easy to use the entropy sampling method when minimizing a log-loss while margin sampling should be used when aiming to reduce classification error.

2.3.5 Active Learning and Multitask Learning

[117] applied AL to MTL for solving linguistic annotations tasks and introduced multitask active learning (MTAL), a framework for multitask and AL. The proposed framework, provides samples to the oracle that are informative for multiple classifiers in the MTL setup

Figure 2.9 Entropy for a Binary Classification



instead of a single one in a traditional AL setup. Thus, creating a corpus labeled in respect to multiple tasks while minimizing the labeling effort provided by AL. The authors used syntactic parsing and NER (highly dissimilar tasks) to test the performance of MTAL and showed that it outperformed random selection and one-sided example selection, where selected samples using AL on one task are used for all other tasks.

MTAL proposed alternating selection and rank combination as sample selection meta-protocols. The alternating selection protocol alternates the task that AL is using. Meaning, a task T_i is used in s_i consecutive AL rounds. Then, another task T_j is selected for s_j rounds and so on. This technique allows to weight annotation coming for each task in the MTL setup. However, [117] only experiment with $s_i = 1$ making this bound to the task sampling method. The rank combination protocol is based on the idea to combine a single task AL selection of all the tasks in the MTL setup. Thus, favoring samples that are informative for all tasks. For each round a informativeness score is calculated for each unlabeled sample x with respect to each task T_i, then converted into a ranking metric $r_{Ti}(x)$, where higher informativeness score mean lower rank value. After that, the rank values from each task are summed to provide a combined rank (Equation 2.14, where it n is the number of tasks).

$$r(x) = \sum_{i=1}^{n} r_{Ti}(x) \tag{2.14}$$

[118] proposed a MTAL framework for SRL following the work of [117]. The framework

uses an Entity Recognition (ER) task as an auxiliary task to reduce the need for extensive training data. The framework uses rank combination as described in Equation 2.14 and the Viterbi decoding (Equation 2.15) to calculate the uncertainty (x_{VE}) in Conditional Random Fields (CRFs) [119] by selecting unlabeled samples with maximum $p_\psi(y|x)$ as defined in Equation 2.16, where x is a sequence of observed vectors, y its corresponding label vectors and $\psi(y_{t-1}, y_t, x_t)$ is the joint log-likelihood function of unary and transition parameters.

$$p_\psi(y|x) = \frac{\prod_{t=1}^{T} \psi(y_{t-1}, y_t, x_t)}{\sum_{y \in Y} \prod_{t=1}^{T} \psi(y_{t-1}, y_t, x_t)} \tag{2.15}$$

$$x_{VE} = \arg\min_x p_\psi(y|x) \tag{2.16}$$

The proposed framework outperformed both single task AL and standard MTL using 12% of data.

2.3.6 Limitations of Active Learning

Most of AL work is conducted in a simulated setup and assume that the oracle is always right, and that labeling queries in not expensive or uniformly expensive which is not always true. [120] studied the lack of a clear stopping criterion since in a real-world application it is hard to monitor the performance at run-time of a classifier trained interactively. Without a performance metric, we are not able to measure the cost induced by labeling (oracle) and misclassification (classifier) costs. Moreover depending on the query strategy calculating the informativeness metric can add computational overhead to the learning algorithm.

An active learner is ultimately a semi-supervised learner so it's prone to harmful biases derived from patterns learned from a small initial labeled dataset.

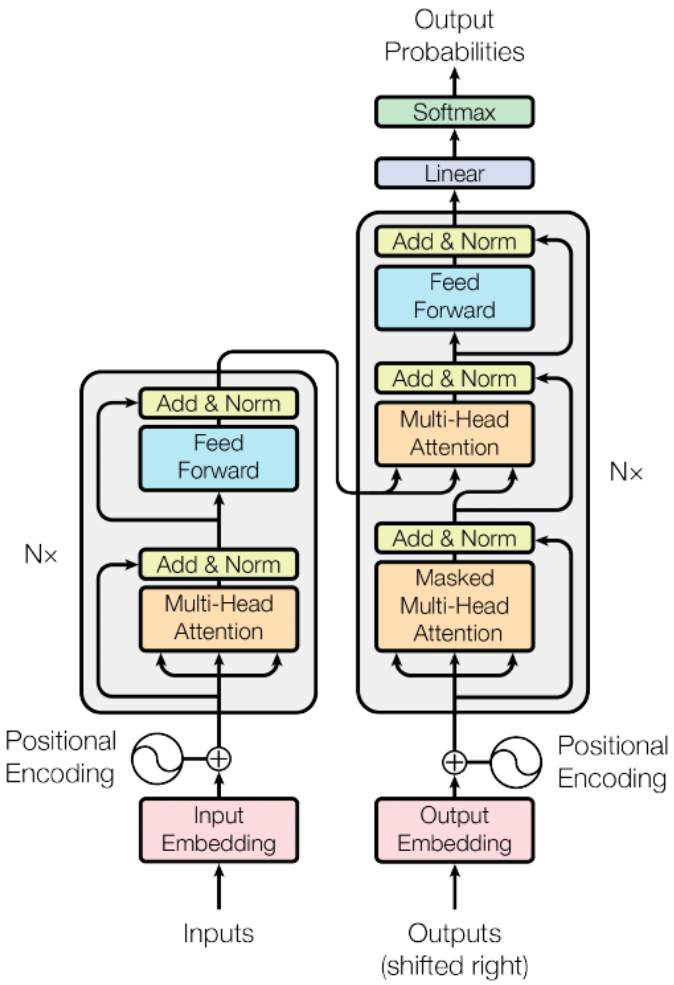
2.4 Transformers

Transformer architectures [36] have gained popularity due to their effectiveness in multiple fields like NLP [121,122] and computer vision [121,123]. Many recent studies focused either on making fundamental architectural changes [124, 125] or improving the efficiency of the initial implementation [126–128] making Transformer based architectures even more competitive.

2.4.1 Architecture

Transformers are formed by stacking multiple identical Transformer blocks. Each block contains an embedding layer, multi-head self-attentions, position-wise fully connected feed-forward networks, layer normalization and residual connectors, for both the encoder and decoder as shown in Figure 2.10. The next sections describe each of these modules following the notation of [36].

Figure 2.10 Transformer architecture as presented in [36]



Encoder

Each encoder block contains two sub-modules, a multi-head self-attention mechanism and a position-wise fully connected feed-forward network, as shown in the left side of Figure 2.10.

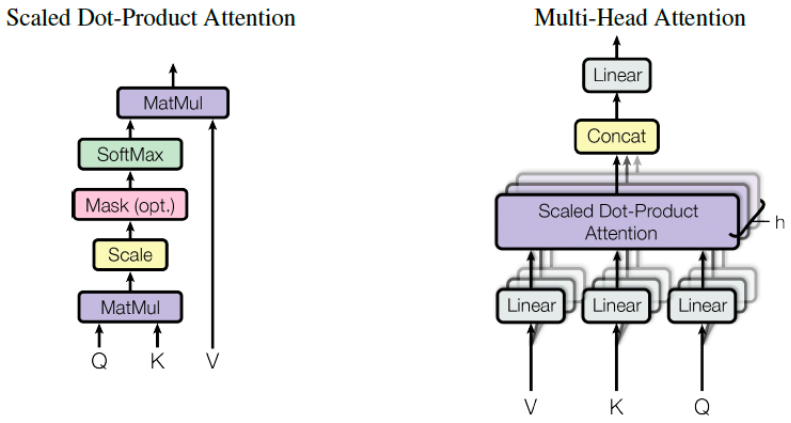
A residual connection followed by layer normalization connects the input and output of each sub-module as shown below, where Sublayer(x) is the sub-module function.

$$\text{LayerNorm}(x + \text{Sublayer}(x)) \tag{2.17}$$

Decoder

The decoder block contains the same sub-modules as the encoder, plus a third multi-head attention sub-module connecting the encoder block as shown in the right side of Figure 2.10. The self-attention sub-module is modified and the output embeddings are offset by one position to guarantee that the prediction of a token depends only on preceding tokens.

Figure 2.11 Scaled Dot-Product Attention (left) and Multi-Head Attention (right) as presented in [36]



Scaled Dot-Product Attention

[36] introduced the scaled dot-product attention mechanism. This attention function calculates the dot products of the query with all the keys similar to dot-product attention, but the result is divided by a scaling factor $\sqrt{d_k}$. In practice, the attention function is calculated in parallel using matrix notation as shown below, where Q (vector representations of each token in the sequence) K (vector representations of all tokens in the sequence) and V are respectively the queries, keys and values matrices.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.18}$$

The weight $w = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})$ describes how each token (Q) attends to all the other tokens K in the sequence. The softmax is used so that the weight value is between 0 and 1.

Multi-Head Self-Attention

The goal of a multi-head self-attention module is to allow the model to attend to different representations of Q, K and V at different positions. The attention function is duplicated with linear projections of Q, K and V. A multi-head attention module can be defined as follows.

$$\text{MutliHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O \tag{2.19}$$

Where $head_i = \text{Attention}(QW_i^Q, KW_i^k, VW_i^V)$. $QW_i^Q, KW_i^k$and VW_i^V are linear projections for Q, K and V respectively, where $W_i^Q, W_i^k$and W_i^V are learned weight matrices.

Position-Wise Fully Connected Feed-Forward Networks

The multi-head attention output is fed to a two-layer position-wise feed-forward network with ReLU activations. It is applied to each position independently. This identical linear transformation can be expressed as follows.

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2.20}$$

Where W_i and b_i are the weight matrix and bias for layer i respectively.

Positional Encoding

The positional encoder injects the token position in the sequence, and it is required since the Transformer architecture doesn't contain recurrence nor convolution. [36] used a sinusoidal input, however, the embeddings can be learned.

2.4.2 Transformer Modes

The Transformer is used in three main ways, correlated to the self-attention usage. (1) Encoder only mode for tasks like classification. In this mode the self-attention modules can attend to any position in the sentence. (2) Decoder only mode for tasks like language modeling. In this mode the self-attention modules can only attend to current and previous positions only in the sentence. (3) Encoder-decoder mode for tasks like machine translation. Like the decode mode, the self-attention modules need to be causal (attending to previous

and present tokens only). This mode also allows the decoder to use information from the input sequence.

2.4.3 Adapters

Adapters are another form of transfer. They're most effective with MLMs that stores rich linguistic information [129–131]. An adapter consists of one or more modules added to a pretrained model. For each task, an adapter adds a few trainable parameters. To maximize the parameter sharing, an adapter architecture usually keeps the adapted model parameters unchanged. Using an adapter, adding a new task doesn't affect previously added tasks.

[37] adapted the BERT Transformer model, showing that an adapter architecture can achieve near SOTA results on 26 NLP tasks, only adding 3.6% parameters per task. Their approach consists of adding a serial adapter after each Transformer sub-module (multi-head self-attention mechanism and the position-wise fully connected feed-forward network). Also, following the work on conditional batch normalization [132], FiLM [133] and self-modulation [134], they trained a different layer normalization for each task. This adapter is connected to the output of each sub-module before the skip connection back. The output of the adapter feeds the next layer normalization. The adapter module follows a bottleneck architecture. First, it projects its d-dimensional features input into m-dimensional features, apply a nonlinearity then projects back to the input dimensions (d), where $m \ll d$. A bottleneck architecture balances between performance and number of added parameters. The module also has a skip-connection. Figure 2.12 shows the architecture proposed by [37].

[38] introduced the Projected Attention Layer (PAL), an adaptation module to a BERT model in a MTL setup and was able to match the performance of fine-tuned models on the GLUE benchmark adding only 1.13x trainable parameters. PAL is a low-dimensional multi-head attention module, with the goal of adapting BERT's original self-attention mechanism to each task. Each PAL is added in parallel to the BERT layers as shown in Figure 2.12, where SA is a self-attention module and LN is a layer-norm in a BERT model.

Figure 2.12 Adapter architecture as presented in [37]

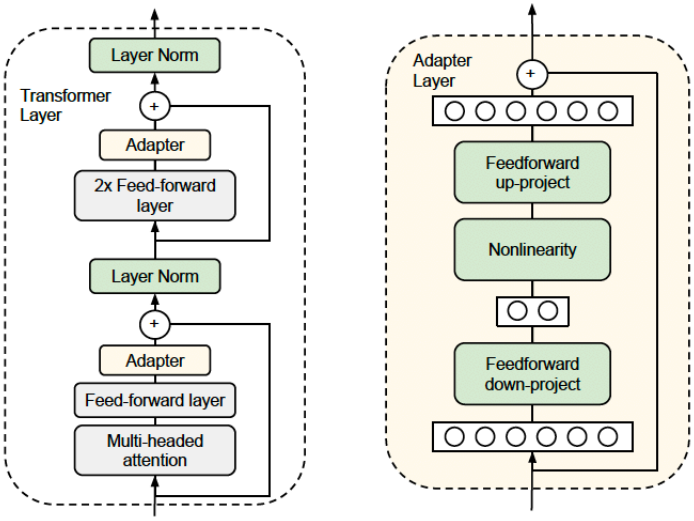
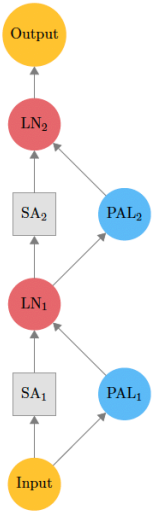


Figure 2.13 Projected Attention Layers' layout as presented in [38]



CHAPTER 3 ARTICLE 1: CONDITIONALLY ADAPTIVE MULTI-TASK LEARNING: IMPROVING TRANSFER LEARNING IN NLP USING FEWER PARAMETERS & LESS DATA

ICLR 2021
Jonathan Pilault, Christopher Pal

3.1 Abstract

Multi-Task Learning (MTL) networks have emerged as a promising method for transferring learned knowledge across different tasks. However, MTL must deal with challenges such as: overfitting to low resource tasks, catastrophic forgetting, and negative task transfer, or learning interference. Often, in Natural Language Processing (NLP), a separate model per task is needed to obtain the best performance. However, many fine-tuning approaches are both parameter inefficient, i.e., potentially involving one new model per task, and highly susceptible to losing knowledge acquired during pretraining. We propose a novel Transformer based Adapter consisting of a new conditional attention mechanism as well as a set of task-conditioned modules that facilitate weight sharing. Through this construction, we achieve more efficient parameter sharing and mitigate forgetting by keeping half of the weights of a pretrained model fixed. We also use a new multi-task data sampling strategy to mitigate the negative effects of data imbalance across tasks. Using this approach, we are able to surpass single task fine-tuning methods while being parameter and data efficient (using around 66% of the data for weight updates). Compared to other BERT Large methods on GLUE, our 8-task model surpasses other Adapter methods by 2.8% and our 24-task model outperforms by 0.7-1.0% models that use MTL and single task fine-tuning. We show that a larger variant of our single multi-task model approach performs competitively across 26 NLP tasks and yields state-of-the-art results on a number of test and development sets. Our code is publicly available at `https://github.com/CAMTL/CA-MTL`.

3.2 Introduction

The introduction of deep, contextualized Masked Language Models (MLM)[1] trained on massive amounts of unlabeled data has led to significant advances across many different Natural Language Processing (NLP) tasks [135,136]. Much of these recent advances can be attributed to the now well-known BERT approach [1]. Substantial improvements over previous state-of-the-art results on the GLUE benchmark [8] have been obtained by multiple groups using BERT models with task specific fine-tuning. The "BERT-variant + fine-tuning" formula

[1]For reader convenience, all acronyms in this paper are summarized in section 3.7.1 of the Appendix.

has continued to improve over time with newer work constantly pushing the state-of-the-art forward on the GLUE benchmark. The use of a single neural architecture for multiple NLP tasks has shown promise long before the current wave of BERT inspired methods [137] and recent work has argued that autoregressive language models (ARLMs) trained on large-scale datasets – such as the GPT family of models [138], are in practice multi-task learners [139]. However, even with MLMs and ARLMs trained for multi-tasking, single task fine-tuning is usually also employed to achieve state-of-the-art performance on specific tasks of interest. Typically this fine-tuning process may entail: creating a task-specific fine-tuned model [1], training specialized model components for task-specific predictions [3] or fine-tuning a single multi-task architecture [4].

Single-task fine-tuning overall pretrained model parameters may have other issues. Recent analyses of such MLM have shed light on the linguistic knowledge that is captured in the hidden states and attention maps [52, 53, 140]. Particularly, BERT has middle Transformer [45] layers that are typically the most transferable to a downstream task [136]. The model proxies the steps of the traditional NLP pipeline in a localizable way [53] — with basic syntactic information appearing earlier in the network, while high-level semantic information appearing in higher-level layers. Since pretraining is usually done on large-scale datasets, it may be useful, for a variety of downstream tasks, to conserve that knowledge. However, single task fine-tuning causes catastrophic forgetting of the knowledge learned during MLM [40]. To preserve knowledge, freezing part of a pretrained network and using *Adapters* for new tasks have shown promising results [3].

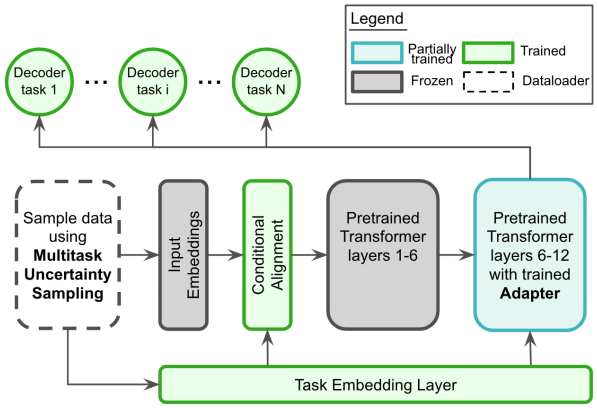


Figure 3.1 CA-MTL base architecture with our uncertainty-based sampling algorithm. Each task has its own decoder. The input embedding layer and the lower Transformer layers are frozen. The upper Transformer layer and Conditional Alignment module are modulated with the task embedding.

Inspired by the human ability to transfer learned knowledge from one task to another new task, Multi-Task Learning (MTL) in a general sense [33, 60, 141] has been applied in many fields outside of NLP. [61] showed that a model trained in a *multi-task* manner can take advantage of the inductive transfer between tasks, achieving a better generalization performance. MTL has the advantage of computational/storage efficiency [142], but training models in a multi-task setting is a balancing act; particularly with datasets that have dif-

ferent: **(a)** dataset sizes, **(b)** task difficulty levels, and **(c)** different types of loss functions. In practice, learning multiple tasks at once is challenging since negative transfer [143], task interference [144, 145] and catastrophic forgetting [146] can lead to worse data efficiency, training stability and generalization compared to single task fine-tuning.

Using Conditionally Adaptive Learning, we seek to improve pretraining knowledge retention and multi-task inductive knowledge transfer. Our contributions are the following:

- A new task conditioned Transformer that adapts and modulates pretrained weights **(Section 3.3.1)**.
- A novel way to prioritize tasks with an uncertainty based multi-task data sampling method that helps balance the sampling of tasks to avoid catastrophic forgetting **(Section 3.3.2)**.

Our Conditionally Adaptive Multi-Task Learning (CA-MTL) approach is illustrated in Figure 3.1. To the best of our knowledge, our work is the first to explore the use of a latent representation of tasks to modularize and adapt pretrained architectures. Further, we believe our work is also the first to examine uncertainty sampling for large-scale multi-task learning in NLP. We show the efficacy of CA-MTL by: **(a)** testing on 26 different tasks and **(b)** presenting state-of-the-art results on a number of test sets as well as superior performance against both single-task and MTL baselines. Moreover, we further demonstrate that our method has advantages over **(c)** other adapter networks, and **(d)** other MTL sampling methods. Finally, we provide ablations and separate analysis of the MT-Uncertainty Sampling technique in section 3.5.1 and of each component of the adapter in 3.5.2.

3.3 Methodology

This section is organized according to the two main MTL problems that we will tackle: (1) How to modularize a pretrained network with latent task representations? (2) How to balance different tasks in MTL? We define each task as: $\mathscr{T}_i \triangleq \{p_i(\mathbf{y}_i|\mathbf{x}_i, \mathbf{z}_i), \mathscr{L}_i, \tilde{p}_i(\mathbf{x}_i)\}$, where $\mathbf{z}_i$ is task i's learnable shallow embedding, $\mathscr{L}_i$ is the task loss, and $\tilde{p}_i(\mathbf{x}_i)$ is the empirical distribution of the training data pair $\{\mathbf{x}_i, \mathbf{y}_i\}$, for $i \in \{1, \ldots, T\}$ and T the number of supervised tasks. The MTL objective is:

$$\min_{\phi(\mathbf{z}), \theta_1, \ldots, \theta_T} \sum_{i=1}^{T} \mathscr{L}_i(f_{\phi(\mathbf{z}_i), \theta_i}(\mathbf{x}_i), \mathbf{y}_i) \tag{3.1}$$

where f is the predictor function (includes encoder model and decoder heads), $\phi(\mathbf{z})$ are learnable generated weights conditioned on $\mathbf{z}$, and θ_i are task-specific parameters for the output decoder heads. $\mathbf{z}$ is constructed using an embedding lookup table.

3.3.1 Task Conditioned Transformer

Our task conditioned Transformer architecture is based on one simple concept. We either add conditional layers or modulate existing pretrained weights using a task representation by extending Feature Wise Linear Modulation [147] functions in several ways depending on the Transformer layer. We define our framework below.

Definition 6 (Conditional Weight Transformations) *Given a neural network weight matrix $\boldsymbol{W}$, we compute transformations of the form $\phi(\boldsymbol{W}|\boldsymbol{z}_i) = \gamma_i(\boldsymbol{z}_i)\,\boldsymbol{W} + \beta_i(\boldsymbol{z}_i)$, where γ_i and β_i are learned functions that transform the weights based on a learned vector embedding $\boldsymbol{z}_i$, for task i.*

Definition 7 (Conditionally Adaptive Learning) *In our setting, Conditionally Adaptive Learning is the process of learning a set of ϕs for the conditionally adaptive modules presented below along with a set of task embedding vectors $\boldsymbol{z}_i$ for T tasks, using a multi-task loss (see equation 3.1).*

In the subsections that follow: We introduce a new Transformer Attention Module using block-diagonal Conditional Attention that allows the original query-key based attention to account for task-specific biases (section **3.3.1**). We propose a new Conditional Alignment method that aligns the data of diverse tasks and that performs better than its unconditioned and higher capacity predecessor (section **3.3.1**). We adapt layer normalization statistics to specific tasks using a new Conditional Layer Normalization module (section **3.3.1**). We add a Conditional Bottleneck that facilitates weight sharing and task-specific information flow from lower layers (section **3.3.1**). In our experiments we provide an ablation study of these components (Table 3.1) examining performance in terms of GLUE scores.

Conditional Attention

Given d, the input dimensions, the query $\mathbf{Q}$, the key $\mathbf{K}$, and the value $\mathbf{V}$ as defined in [45], we redefine the attention operation:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{z}_i)) = \text{softmax}\left[M(\mathbf{z}_i) + \frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right]\mathbf{V}$$

$$M(\mathbf{z}_i) = \bigoplus_{n=1}^{N} A'_n(\mathbf{z}_i), \quad A'_n(\mathbf{z}_i) = A_n\gamma_i(\mathbf{z}_i) + \beta_i(\mathbf{z}_i)$$

where $\bigoplus$ is the direct sum operator (see section 3.7.6), N is the number of block matrices $A_n \in \mathbb{R}^{(L/N)\times(L/N)}$ along the diagonal of the attention matrix, L is the input sequence, $M(\mathbf{z}_i) = \text{diag}(A'_1, \ldots, A'_N)$ is a block diagonal conditional matrix. Note that A_n is constructed using L/N trainable and randomly initialized L/N dimensional vectors. While the original attention matrix depends on the hidden states h, $M(\mathbf{z}_i)$ is a learnable weight matrix that only depends on the task embedding $\mathbf{z}_i \in \mathbb{R}^d$. $\gamma_i, \beta_i : \mathbb{R}^d \mapsto \mathbb{R}^{L^2/N^2}$ are Feature Wise Linear Modulation [147] functions. We also experimented with full-block Conditional Attention $\in \mathbb{R}^{L\times L}$. Not only did it have N^2 more parameters compared to the block-diagonal variant, but it also performed significantly worse on the GLUE development set (see FBA variant in Table 3.10). It is possible that GLUE tasks derive a certain benefit from localized attention that is a consequence of $M(\mathbf{z}_i)$. With $M(\mathbf{z}_i)$, each element in a sequence can only attend to other elements in its subsequence of length L/N. In our experiments we used $N = d/L$. The full Conditional Attention mechanism used in our experiments is illustrated in Figure 3.2.

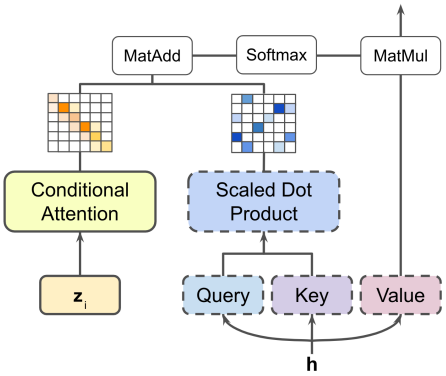


Figure 3.2 Conditional Attention Module

Conditional Alignment

[144] showed that in MTL having T separate alignment modules $R_1, \ldots, R_T$ increases $\text{BERT}_{\text{LARGE}}$ avg. scores on five GLUE tasks (CoLA, MRPC, QNLI, RTE, SST-2) by 2.35%. Inspired by this work, we found that adding a task conditioned alignment layer between the input embedding layer and the first BERT Transformer layer improved multi-task model performance. However, instead of having T separate alignment matrices R_i for each T task, one alignment matrix $\hat{R}$ is generated as a function of the task embedding z_i. As in [144], we tested this module on the same five GLUE tasks and with $\text{BERT}_{\text{LARGE}}$. Enabling task conditioned weight sharing across covariance alignment modules allows us to outperforms $\text{BERT}_{\text{LARGE}}$ by 3.61%. This is 1.26 % higher than having T separate alignment matrices. Inserting $\hat{R}$ into BERT, yields the following encoder function $\hat{f}$:

$$\hat{f} = \sum_{t=1}^{T} g_{\theta_i}(E(\mathbf{x}_i)\hat{R}(\mathbf{z}_i)B), \qquad \hat{R}(\mathbf{z}_i) = R\gamma_i(\mathbf{z}_i) + \beta_i(\mathbf{z}_i) \tag{3.2}$$

where $\mathbf{x}_i \in \mathbb{R}^d$ is the layer input, g_{θ_i} is the decoder head function for task i with weights θ_i, E the frozen BERT embedding layer, B the BERT Transformer layers and R the linear

weight matrix of a single task conditioned alignment matrix. $\gamma_i, \beta_i : \mathbb{R}^d \mapsto \mathbb{R}^d$ are Feature Wise Linear Modulation functions.

Conditional Layer Normalization (CLN)

We extend the Conditional Batch Normalization idea from [148] to Layer Normalization [149]. For task $\mathscr{T}_i$, $i \in \{1, \ldots, T\}$:

$$\mathbf{h}_i = \frac{1}{\sigma} \odot (\mathbf{a}_i - \mu) * \hat{\gamma}_i(\mathbf{z}_i) + \beta_i(\mathbf{z}_i), \qquad \hat{\gamma}_i(\mathbf{z}_i) = \boldsymbol{\gamma}' \gamma_i(\mathbf{z}_i) + \boldsymbol{\beta}' \tag{3.3}$$

where $\mathbf{h}_i$ is the CLN output vector, $\mathbf{a}_i$ are the preceding layer activations associated with task i, μ and σ are the mean and the variance of the summed inputs within each layer as defined in [149]. Conditional Layer Normalization is initialized with BERT's Layer Normalization affine transformation weights and bias $\boldsymbol{\gamma}'$ and $\boldsymbol{\beta}'$ from the original formulation: $\mathbf{h} = \frac{1}{\sigma} \odot (\mathbf{a} - \mu) * \boldsymbol{\gamma}' + \boldsymbol{\beta}'$. During training, the weight and bias functions of $\gamma_i(*)$ and $\beta_i(*)$ are always trained, while the original Layer Normalization weight may be kept fixed. This module was added to account for task specific rescaling of individual training cases. Layer Normalization normalizes the inputs across features. The conditioning introduced in equation 3.3.1 allows us to modulate the normalization's output based on a task's latent representation.

Conditional Bottleneck

We created a task conditioned two layer feed-forward bottleneck layer (CFF up/down in Figure 3.3). The conditional bottleneck layer follows the same transformation as in equation 3.2. The module in Figure 3.3a is added to the top most Transformer layers of CA-MTL$_{\text{BASE}}$ and uses a CLN. For CA-MTL$_{\text{LARGE}}$ this module is the main building block of the skip connection added alongside all Transformer layers seen in Figure 3.3b. The connection at layer j takes in the matrix sum of the Transformer layer output at j and the previous connection's output at $j-1$. The Conditional bottleneck allows lower layer information to flow upwards depending on the task. Our intuition for introducing this component is related to recent studies [53] that showed that the "most important layers for a given task appear at specific positions". As with the other modules

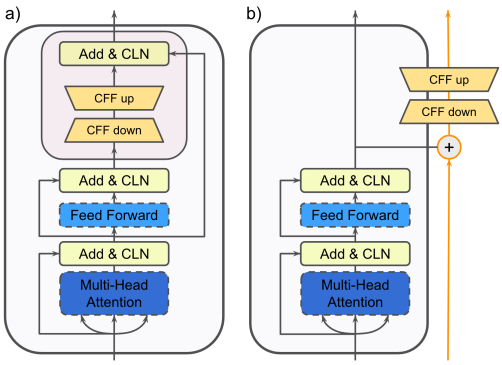


Figure 3.3 a) Conditional Bottleneck for CA-MTL$_{\text{BASE}}$. b) Conditional Bottleneck for CA-MTL$_{\text{LARGE}}$.

described so far, each task adaptation is created from the weights of a single shared adapter that is modulated by the task embedding.

3.3.2 Multi-Task Uncertainty Sampling

MT-Uncertainty Sampling is a task selection strategy that is inspired by Active Learning techniques. Our algorithm 2 is outlined in the Appendix, Section 3.7.2. Similar to Active Learning, our algorithm first evaluates the model uncertainty. MT-Uncertainty Sampling uses Shannon Entropy, an uncertainty measure, to choose training examples by first doing forward pass through the model with $b \times T$ input samples. For an output classification prediction with C_i possible classes and probabilities $(p_{i,1}, \ldots, p_{i,C_i})$, the Shannon Entropy H_i, for task $\mathscr{T}_i$ and $i \in \{1, \ldots, T\}$, our uncertainty measure $\mathscr{U}(\mathrm{x})$ are given by:

$$H_i = H_i(f_{\phi(\mathbf{z}_i),\theta_i}(\mathrm{x})) = -\sum_{c=1}^{C_i} p_c \log p_c, \qquad \mathscr{U}(x_i) = \frac{H_i(f_{\phi(\mathbf{z}_i),\theta_i}(\mathrm{x}))}{\hat{H} \times H_i'} \tag{3.4}$$

$$\hat{H} = \max_{i \in \{1,\ldots,T\}} \bar{H}_i = \max\left[\frac{1}{b}\sum_{\mathrm{x} \in \mathbf{x}_i} H_i\right], \qquad H_i' = -\sum_{c=1}^{C_i} \frac{1}{C_i} \log\left[\frac{1}{C_i}\right] \tag{3.5}$$

where $\bar{H}_i$ is the average Shannon Entropy across b samples of task t, H_i', the Shannon entropy of choosing classes with uniform distribution and $\hat{H}$, the maximum of each task's average entropy over b samples. H_i' is normalizing factor that accounts for differing number of prediction classes (without the normalizing factor H_i', tasks with a binary classification $C_i = 1$ were rarely chosen). Further, to limit high entropy outliers and to favor tasks with highest uncertainty, we normalize with $\hat{H}$. The measure in eq. 3.4 allows Algorithm 2 to choose b samples from $b \times T$ candidates to train the model.

3.4 Related Work

Multi-Tasking in NLP. To take advantage of the potential positive transfer of knowledge from one task to another, several works have proposed carefully choosing which tasks to train as an intermediate step in NLP before single task fine-tuning [6, 143, 150–153]. The intermediate tasks are not required to perform well and are not typically evaluated jointly. In this work, all tasks are trained *jointly* and *all tasks used* are evaluated from a *single model*. In Natural Language Understanding (NLU), it is still the case that to get the best task performance one often needs a separate model per task [154,155]. At scale, Multilingual NMT systems [156] have also found that MTL model performance degrades as the number of

tasks increases. We notice a similar trend in NLU with our baseline MTL model. Recently, approaches in MTL have tackled the problem by designing task specific decoders on top of a shared model [4] or distilling multiple single-task models into one [154]. Nonetheless, such MTL approaches still involves single task fine-tuning. In this paper, we show that it is possible to achieve high performance in NLU without single task fine-tuning.

Adapters. Adapters are trainable modules that are attached in specific locations of a pretrained network. They provide another promising avenue to limit the number of parameters needed when confronted with a large number of tasks. This approach is useful with pretrained MLM models that have rich linguistic information [52, 53, 136, 157]. Recently, [3] added an adapter to a pretrained BERT model by fine-tuning the layer norms and adding feed forward bottlenecks in every Transformer layer. However, such methods adapt each task individually during the fine-tuning process. Unlike prior work, our method harnesses the vectorized representations of tasks to modularize a single pretrained model across all tasks. [2] and [158] also mix both MTL and adapters with BERT and T5 encoder-decoder [159] respectively by creating local task modules that are controlled by a global task agnostic module. The main drawback is that a new set of non-shared parameters must be added when a new task is introduced. CA-MTL shares all parameters and is able to re-modulate existing weights with a new task embedding vector.

Active Learning, Task Selection and Sampling. Our sampling technique is similar to the ones found in several active learning algorithms [160] that are based on Shannon entropy estimations. [161] and [162] examined Multi-Task Active Learning (MTAL), a technique that chooses one informative sample for T different learners (or models) for each T tasks. Instead we choose T tasks samples for *one model*. Moreover, the algorithm weights each sample by the corresponding task score, and the Shannon entropy is normalized to account for various losses (see equation 3.5). Also, our algorithm is used in a large scale MTL setup ($\gg 2$ tasks). Recently, [39] explored task selection in MTL using learning policies based on counterfactual estimations [163]. However, such method considers only fixed stochastic parameterized policies while our method *adapts* its selection criterion based on model uncertainty throughout the training process.

3.5 Experiments and Results

We show that our adapter of section 3.3 achieve parameter efficient transfer for 26 NLP tasks. Our implementation of CA-MTL is based on HuggingFace [164]. Hyperparameters and our experimental set-up are outlined in 3.7.5. To preserve the weights of the pretrained model, CA-MTL's bottom half Transformer layers are frozen in all experiments (except in section

3.5.4). We also tested different layer freezing configurations and found that freezing half the layers worked best on average (see Section 3.7.8).

3.5.1 Multi-Task Uncertainty Sampling

Our MT-Uncertainty sampling strategy, from section 3.3.2, is compared to 3 other task selection schemes: a) Counterfactual b) Task size c) Random. We used a $\text{BERT}_{\text{BASE}}$ (no adapters) on 200k iterations and with the same hyperparameters as in [39]. For more information on Counterfactual task selection, we invite the reader to consult the full explanation in [39]. For T tasks and the dataset D_i for tasks $i \in \{1, \ldots, T\}$, we rewrite the definitions of Random π_{rand} and Task size $\pi_{|task|}$ sampling:

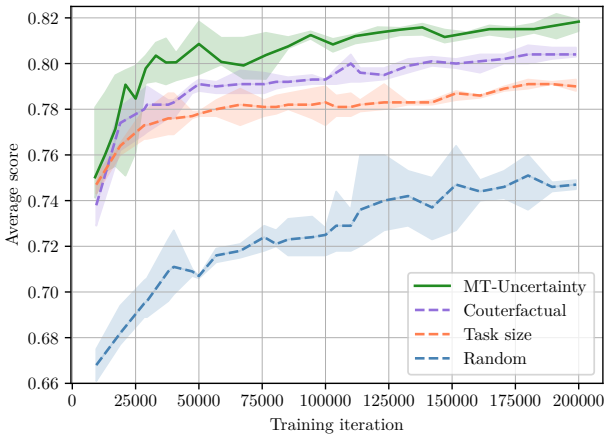


Figure 3.4 **MT-Uncertainty** vs. other task sampling strategies: median dev set scores on 8 GLUE tasks and using $\text{BERT}_{\text{BASE}}$. Data for the Counterfactual and Task Size policy $\pi_{|task|}$ (eq. 3.6) is from [39].

$$\pi_{rand} = 1/T, \quad \pi_{|task|} = |D_i| \left[\sum_{i=1}^{T} |D_i| \right]^{-1} \tag{3.6}$$

In Figure 3.4, we see from the results that MT-Uncertainty converges faster by reaching the 80% average GLUE score line before other task sampling methods. Further, MT-Uncertainty maximum score on 200k iterations is at 82.2, which is 1.7% higher than Counterfactual sampling. The datasets in the GLUE benchmark offers a wide range of dataset sizes. This is useful to test how MT-Uncertainty manages a jointly trained low resource task (CoLA) and high resource task (MNLI). Figure 3.5 explains how catastrophic forgetting is curtailed by sampling tasks before performance drops. With π_{rand}, all of CoLA's tasks are sampled by

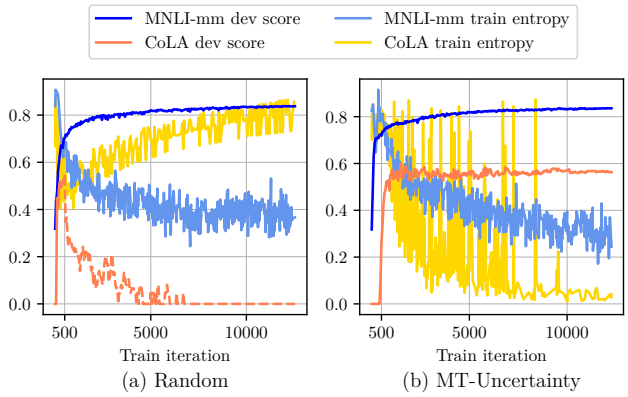


Figure 3.5 CoLA/MNLI Dev set scores and Entropy for π_{rand} (left) and **MT-Uncertainty** (right).

iteration 500, at which point the larger MNLI dataset overtakes the learning process and CoLA's dev set performance starts to diminish. On the other hand, with MT-Uncertainty sampling, CoLA is sampled whenever Shannon entropy is higher than MNLI's. The model first assesses uncertain samples using Shannon Entropy then decides what data is necessary to train on. This process allows lower resource tasks to keep performance steady. We provide evidence in Figure 3.8 of 3.7.2 that MT-Uncertainty is able to manage task difficulty — by choosing the most difficult tasks first.

3.5.2 Ablation and Module Analysis

In Table 3.1, we present the results of an ablation study to determine which elements of CA-MTL$_{\text{BERT-BASE}}$ had the largest positive gain on average GLUE scores. Starting from a MTL BERT$_{\text{BASE}}$ baseline trained using random task sampling (π_{rand}). Apart for the Conditional Adapter, each module as well as MT-Uncertainty lift overall performance and reduce variance across tasks. Please note that we also included accuracy/F1 scores for QQP, MRPC and Pearson/ Spearman correlation for STS-B to calculate score standard deviation Task σ. Intuitively, when negative task transfer occurs between two tasks, either (1) task interference is bidirectional and scores are both impacted, or (2) interference is unidirectional and only one score is impacted.

Table 3.1 Model ablation study[a] on the GLUE dev set. All models have the bottom half layers frozen.

Model changes	Avg GLUE	Task σ GLUE	% data used
BERT$_{\text{BASE}}$ MTL (π_{rand})	80.61	14.41	100
+ Conditional Attention	82.41	10.67	100
+ Conditional Adapter	82.90	11.27	100
+ CA and CLN	83.12	10.91	100
+ MT-Uncertainty (CA-MTL$_{\text{BERT-BASE}}$)	**84.03**	**10.02**	66.3

[a]CA=Conditional Alignment, CLN=Conditional Layer Normalization, Task σ=scores standard deviation *across tasks*.

We calculate Task σ to characterize changes in the dynamic range of performance across multiple tasks. We do this to asses the degree to which performance improvements are distributed across all tasks or only subsets of tasks. As we can see from Table 3.1, Conditional Attention, Conditional Alignment, Conditional Layer Normalization, MT-Uncertainty play roles in reducing

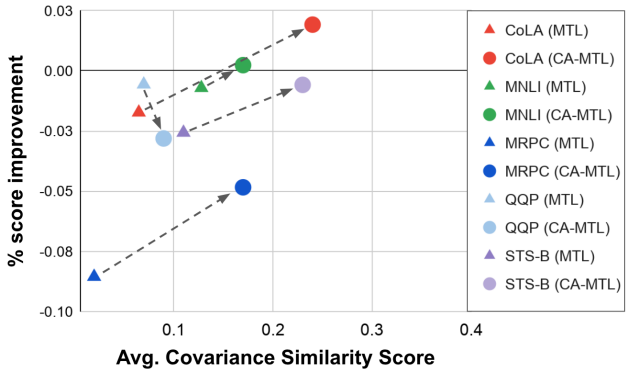


Figure 3.6 Task performance vs. avg. covariance similarity scores (eq. 3.7) for MTL and CA-MTL.

Task σ and increasing performance across tasks. This provides partial evidence of CA-MTL's ability to mitigating negative task transfer. We show that Conditional Alignment can learn to capture covariate distribution differences with task embeddings co-learned from other adapter components of CA-MTL. In Figure 3.6, we arrive at similar conclusions as [144], who proved that negative task transfer is reduced when task covariances are aligned. The authors provided a "covariance similarity score" to gauge covariance alignment. For task i and j with m_i and m_j data samples respectively, and given d dimensional inputs to the first Transformer layer $X_i \in \mathbb{R}^{m_i \times d}$ and $X_j \in \mathbb{R}^{m_j \times d}$, we rewrite the steps to calculate the covariance similarity score between task i and j: (a) Take the covariance matrix $X_i^\top X_i$, (b) Find its best rank-r_i approximation $U_{i,r_i} D_{i,r_i} U_{i,r_i}^\top$, where r_i is chosen to contain 99% of the singular values. (c) Apply steps (a), (b) to X_j, and compute the covariance similarity score $CovSim_{i,j}$:

$$CovSim_{i,j} := \frac{\|(U_{i,r_i} D_{i,r_i}^{1/2})^\top U_{j,r_j} D_{j,r_j}^{1/2}\|_F}{\|U_{i,r_i} D_{i,r_i}^{1/2}\|_F \cdot \|U_{j,r_j} D_{j,r_j}^{1/2}\|_F}. \;\; CovSim_i = \frac{1}{T-1} \sum_{j \neq i} CovSim_{i,j} \tag{3.7}$$

Since we are training models with T tasks, we take the average covariance similarity score $CovSim_i$ between task i and all other tasks. We measure $CovSim_i$ using equation 3.7 between 9 single-task models trained on individual GLUE tasks. For each task in Figure 3.6, we measure the similarity score on the MTL trained $\text{BERT}_{\text{BASE}}$ baseline, e.g., CoLA (MTL), or $\text{CA-MTL}_{\text{BERT-BASE}}$ model, e.g., MNLI (CA-MTL). Our score improvement measure is the % difference between a single task model and MTL or CA-MTL on the particular task. We find that covariance similarity increases for 9 tasks and that performance increases for 7 out 9 tasks. These measurements confirm that the Conditional Alignment is able to align task covariance, thereby helping alleviate task interference.

3.5.3 Jointly training on 8 tasks: GLUE

In Table 3.2, we evaluate the performance of CA-MTL against single task fine-tuned models, MTL as well as the other BERT-based adapters on GLUE. As in [3], MNLI_m and MNLI_mm are treated as separate tasks. Our results indicate that CA-MTL outperforms both the BASE adapter, PALS+Anneal Sampling [2], and the LARGE adapter, Adapters-256 [3]. Against single task (ST) models, CA-MTL is 1.3% higher than $\text{BERT}_{\text{BASE}}$, with 5 out 9 tasks equal or greater performance, and 0.7% higher than $\text{BERT}_{\text{LARGE}}$, with 3 out 9 tasks equal or greater performance. ST models, however, need 9 models or close to 9× more parameters for all 9 tasks. We noted that $\text{CA-MTL}_{\text{BERT-LARGE}}$'s average score is driven by strong RTE scores. While RTE benefits from MTL, this behavior may also be a side effect of layer freezing. In

Table 3.2 Adapters with layer freezing vs. ST/MT on GLUE test set. F1 scores are reported for QQP/MRPC, Spearman's correlation for STS-B, accuracy on the matched/mismatch sets for MNLI, Matthew's correlation for CoLA and accuracy for other tasks. * Individual scores not available. ST=Single Task, MTL=Multitask, g.e.= greater or equal to. Results from: [1] [1] [2] [2]. [3] [3] .

Method	Type	Total params	Trained params/task	# tasks g.e. ST	GLUE CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	Avg
Base Models — Test Server Results													
$\text{BERT}_{\text{BASE}}$[1]	ST	9.0×	100%	—	52.1	84.6/83.4	88.9	90.5	71.2	66.4	93.5	85.8	79.6
$\text{BERT}_{\text{BASE}}$[2]	MTL	1.0×	11.1%	2	51.2	84.0/83.4	86.7	89.3	70.8	76.6	93.4	83.6	79.9
PALs+Anneal Samp.[2]	MTL	1.13×	12.5%	4	51.2	84.3/83.5	88.7	90.0	71.5	76.0	92.6	85.8	80.4
CA-MTL$_{\text{BERT-BASE}}$ (ours)	MTL	1.12×	5.6 %	**5**	53.1	85.9/85.8	88.6	90.5	69.2	76.4	93.2	85.3	**80.9**
Large Models — Test Server Results													
$\text{BERT}_{\text{LARGE}}$[1]	ST	9.0×	100%	—	60.5	86.7/85.9	89.3	92.7	72.1	70.1	94.9	86.5	82.1
Adapters-256[3]	ST	1.3×	3.6%	3	59.5	84.9/85.1	89.5	90.7	71.8	71.5	94.0	86.9	80.0
CA-MTL$_{\text{BERT-LARGE}}$ (ours)	MTL	1.12×	5.6%	3	59.5	85.9/85.4	89.3	92.6	71.4	79.0	94.7	87.7	**82.8**

Table 3.10, we see that CA-MTL has gains over ST on more and more tasks as we gradually unfreeze layers.

3.5.4 Transfer to New Tasks

In Table 3.3 we examine the ability of our method to quickly adapt to new tasks. We performed domain adaptation on SciTail [30] and SNLI [29] datasets, using a $\text{CA-MTL}_{\text{BASE}}$ model trained on GLUE and a new linear decoder head. We tested several pre-trained and randomly initialized task embeddings in a zero-shot setting. The complete set of experiments with all task embeddings can be found in the Appendix, Section 3.7.4. We then selected the best task embedding for our results in Table 3.3. STS-B and MRPC MTL-trained task embeddings performed best on SciTail and SNLI respectively. $\text{CA-MTL}_{\text{BERT-BASE}}$ has faster adaptation than $\text{MT-DNN}_{\text{SMART}}$ [5] as evidenced by higher performances in low-resource regimes (0.1% and 1% of the data). When trained on the complete dataset, $\text{CA-MTL}_{\text{BERT-BASE}}$ is on par with $\text{MT-DNN}_{\text{SMART}}$. Unlike $\text{MT-DNN}_{\text{SMART}}$ however, we do not add context from a semantic similarity model – $\text{MT-DNN}_{\text{SMART}}$ is built off HNN [165]. Nonetheless, with a larger model, CA-MTL surpasses $\text{MT-DNN}_{\text{SMART}}$ on the full SNLI and SciTail datasets in Table 3.6.

Table 3.3 Domain adaptation results on dev. sets for *BASE* models. [1] [4], [2] [5]

% data used	SciTail 0.1%	SciTail 1%	SciTail 10%	SciTail 100%	SNLI 0.1%	SNLI 1%	SNLI 10%	SNLI 100%
$\text{BERT}_{\text{BASE}}$[1]	51.2	82.2	90.5	94.3	52.5	78.1	86.7	91.0
MT-DNN[1]	81.9	88.3	91.1	95.7	81.9	88.3	91.1	95.7
$\text{MT-DNN}_{\text{SMART}}$[2]	82.3	88.6	91.3	**96.1**	82.7	86.0	**88.7**	**91.6**
$\text{CA-MTL}_{\text{BERT}}$	**83.2**	**88.7**	**91.4**	95.6	**82.8**	**86.2**	88.0	91.5

3.5.5 Jointly training on 24 tasks: GLUE/Super-GLUE, MRQA and WNUT2017

Effects of Scaling Task Count. In Figure 3.7 we continue to test if CA-MTL mitigates *task interference* by measuring GLUE average scores when progressively adding 9 GLUE tasks, 8 Super-GLUE tasks [15], 6 MRQA tasks [22]. Tasks are described in Appendix section 3.7.9. The results show that adding 23 tasks drops the performance of our baseline MTL $\text{BERT}_{\text{BASE}}$ (π_{rand}).

MTL BERT increases by 4.3% when adding MRQA but, with 23 tasks, the model performance drops by 1.8%. The opposite is true when CA-MTL modules are integrated into the model. CA-MTL continues to show gains with a large number of tasks and surpasses the baseline MTL model by close to 4% when trained on 23 tasks.

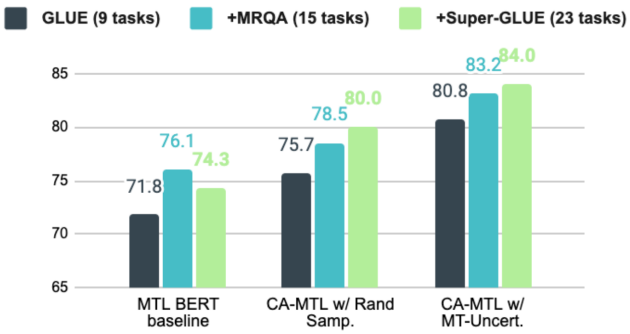


Figure 3.7 Effects of adding more datasets on avg GLUE scores. Experiments conducted on 3 epochs. When 23 tasks are trained jointly, performance of $\text{CA-MTL}_{\text{BERT-BASE}}$ continues to improve.

24-task CA-MTL. We *jointly* trained large MTL baselines and CA-MTL models on GLUE/Super-GLUE/MRQA and Named Entity Recognition (NER) WNUT2017 [166]. Since some dev. set scores are not provided and since RoBERTa results were reported with a median score over 5 random seeds, we ran our own single seed ST/MTL baselines (marked "ReImp") for a fair comparison. The dev. set numbers reported in [7] are displayed with our baselines in Table 3.9. Results are presented in Table 3.4. We notice in Table 3.4 that even for large models, CA-MTL provides large gains in performance on average over both ST and MTL models. For the BERT based models, CA-MTL provides 2.3% gain over ST and higher scores on 17 out 24 tasks. For RoBERTa based models, CA-MTL provides 1.2% gain over ST and higher scores on 15 out 24 tasks. We remind the reader that this is achieved with a

Table 3.4 24-task CA-MTL vs. ST and vs. 24-task MTL with frozen layers on GLUE, SuperGLUE, MRQA and NER development sets. ST=Single Task, MTL=Multitask, g.e.= greater or equal to. Details in section 3.7.5.

Model	Task Grouping				Avg	# tasks e.g. ST	Total Params
	GLUE	SuperGLUE	MRQA	NER			
BERT-LARGE models							
ST_{ReImp}	84.5	68.9	79.7	54.1	76.8	—	24×
$\text{MTL}_{\text{ReImp}}$	83.2	72.1	77.8	42.2	76.4	9/24	1×
CA-MTL	86.6	74.1	79.5	49.0	79.1	**17/24**	1.12×
RoBERTa-LARGE models							
ST_{ReImp}	88.2	76.5	83.6	57.8	81.9	—	24×
$\text{MTL}_{\text{ReImp}}$	86.0	78.6	80.7	49.3	80.7	7/24	1×
CA-MTL	89.4	80.0	82.4	55.2	83.1	**15/24**	1.12×

single model. Even when trained with 16 other tasks, it is interesting to note that the MTL baseline perform better than the ST baseline on Super GLUE where most tasks have a small number of samples. Also, we used NER to test if we could still outperform the ST baseline on a token-level task, significantly different from other tasks. Unfortunately, while CA-MTL performs significantly better than the MTL baseline model, CA-MTL had not yet overfit on this particular task and could have closed the gap with the ST baselines with more training cycles.

Comparisons with other methods. In Table 3.5, CA-MTL$_{\text{BERT}}$ is compared to other Large BERT based methods that either use MTL + ST, such as MT-DNN [4], intermediate tasks + ST, such as STILTS [6] or MTL model distillation + ST, such as BAM! [154]. Our method scores higher than MT-DNN on 5 of 9 tasks and by 1.0 % on avg. Against STILTS, CA-MTL realizes a 0.7 % avg. score gain, surpassing scores on 6 of 9 tasks. We also show that CA-MTL$_{\text{RoBERTa}}$ is within only 1.6 % of a RoBERTa ensemble of 5 to 7 models per task and that uses intermediate tasks. Using our 24-task CA-MTL large RoBERTa-based model, we report NER F1 scores on the WNUT2017 test set in Table 3.6a. We compare our result with RoBERTa$_{\text{LARGE}}$ and XLM-R$_{\text{LARGE}}$ [167] the current state-of-the-art (SOTA). Our model outperforms XLM-R$_{\text{LARGE}}$ by 1.6%, reaching a new state-of-the-art. Using domain adaptation as described in Section 3.5.4, we report results on the SciTail test set in Table 3.6b and SNLI test set in Table 3.6c. For SciTail, our model matches the current SOTA[2] ALUM [168], a RoBERTa large based model that additionally uses the SMART [5] fine-tuning method. For SNLI, our model outperforms SemBert, the current SOTA[3].

Table 3.5 Our 24-task CA-MTL vs. other large models on GLUE. F1 is reported for QQP/MRPC, Spearman's corr. for STS-B, Matthew's corr. for CoLA and accuracy for other tasks. *Split not available. **Uses intermediate task fine-tuning + ST.

Model	GLUE tasks								Avg
	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	
BERT-LARGE based models on Dev set.									
MT-DNN	63.5	87.1/86.7	91.0	92.9	89.2	83.4	94.3	90.6	85.6
STILTS **	62.1	86.1*	92.3	90.5	88.5	83.4	93.2	90.8	85.9
BAM!	61.8	87.0*	–	92.5	–	82.8	93.6	89.7	–
24-task CA-MTL	63.8	86.3/86.0	92.9	93.4	88.1	84.5	94.5	90.3	**86.6**
RoBERTa-LARGE based models on Test set.									
RoBERTA** with Ensemble	67.8	91.0/90.8	91.6	95.4	74.0	87.9	97.5	92.5	**87.3**
24-task CA-MTL	62.2	89.0/88.4	92.0	94.7	72.3	86.2	96.3	89.8	85.7

[2]https://leaderboard.allenai.org/scitail/submissions/public on 09/27/2020
[3]https://nlp.stanford.edu/projects/snli/ on 09/27/2020

Table 3.6 CA-MTL test performance vs. SOTA.

(a) WNUT2017	F1
RoBERTa$_{\text{LARGE}}$	56.9
XLM-R$_{\text{LARGE}}$	57.1
CA-MTL$_{\text{RoBERTa}}$ (ours)	**58.0**

(b) SciTail	% Acc
MT-DNN	94.1
ALUM$_{\text{RoBERTa}}$	96.3
ALUM$_{\text{RoBERTa-SMART}}$	**96.8**
CA-MTL$_{\text{RoBERTa}}$ (ours)	**96.8**

(c) SNLI	% Acc
MT-DNN	91.6
MT-DNN$_{\text{SMART}}$	91.7
SemBERT	91.9
CA-MTL$_{\text{RoBERTa}}$ (ours)	**92.1**

3.6 Conclusion

We believe that our experiments here have helped demonstrate the potential of task conditioned adaptive learning within a single model that performs multiple tasks. In a large-scale 24-task NLP experiment, CA-MTL outperforms fully tuned single task models by 2.3% for BERT Large and by 1.2% for RoBERTa Large using 1.12 times the number of parameters, while single task fine-tuning approach requires 24 separately tuned single task models or 24 times the number of parameters. When a BERT vanilla MTL model sees its performance drop as the number of tasks increases, CA-MTL scores continue to climb. Performance gains are not driven by a single task as it is often the case in MTL. Each CA-MTL module that adapts a Transformer model is able to reduce performance variances between tasks, increasing average scores and aligning task covariances. This evidence shows that CA-MTL is able to mitigate task interference and promote more efficient parameter sharing. We showed that MT-Uncertainty is able to avoid degrading performances of low resource tasks. Tasks are sampled whenever the model sees entropy increase, helping avoid catastrophic forgetting. Overall, CA-MTL offers a promising avenue to dynamically adapt and modularize knowledge embedded in large monolithic pretrained models. Extending such ideas will be an objective for future work.

Acknowledgments

This research was supported by the Canada CIFAR AI Chairs Program, NSERC and PROMPT. Experiments in this article were conducted with Compute Canada and MILA computational infrastructure and we thank them for their support. We would like to thank Colin Raffel, Sandeep Subramanian, and Nicolas Gontier for their useful feedback and the anonymous reviewers for helpful comments, discussions and suggestions.

3.7 Appendix from ICLR 2021 Paper

3.7.1 Summary of Acronyms

Acronyms of datasets and descriptions can be found below in section 3.7.9.

Table 3.7 List of acronyms used in this paper.

Acronym	Description
ARLM	Autoregressive Language Models
CA-MTL	Conditional Adaptive Multi-Task Learning: our architecture
CFF	Conditional Feed-Forward: a feed-forward layer modulated by a conditioning vector
CLN	Conditional Layer Normalization in section 3.3.1
EDM	Evolutionary Data Measures [169]: a task difficulty estimate
GLUE	General Language Understanding Evaluation [8]: a benchmark with multiple datasets
QA	Question Answering
MT	Multi-Task
MTAL	Multi-Task Active Learning: finding the most informative instance for multiple learners (or models)
MLM	Masked Language Model: BERT [1] is an example of an MLM
MTL	Multi-Task Learning: "learning tasks in parallel while using a shared representation" [60]
MRQA	Machine Reading for Question Answering [22]: a benchmark with multiple datasets
NER	Named Entity Recognition
NLP	Natural Language Processing
SOTA	State of the art
ST	Single Task fine-tuning: all weights are typically updated
ST-A	ST with Adapter modules: one adapter per task is trained and pretrained weights are optionally updated

3.7.2 Uncertainty Sampling: Algorithm and Additional Results

Algorithm 2: Multi-task Uncertainty Sampling

Input: Training data D_t for task $t \in [1, \dots, T]$; batch size b; C_t possible output classes for task t; $f := f_{\phi(\mathbf{z}_i),\theta_i}$ our model with weights ϕ, θ_i;

Output: $\mathcal{B}'$ - multi-task batch of size b

$\mathcal{B} \leftarrow \emptyset$

for $t \leftarrow 1$ **to** T **do**

  Generate $\mathbf{x}_t := \{x_{t,1}, \dots, x_{t,b}\} \overset{\text{i.i.d.}}{\sim} D_t$

  for $i \leftarrow 1$ **to** b **do**

    $\mathcal{H}_{t,i} \leftarrow -\sum_{c=1}^{C_t} p_c(f(x_{t,i}))\ \log\ p_c(f(x_{t,i}))$ ▷ Entropy of each sample

  end

  Compute $\bar{\mathcal{H}}_t \leftarrow \frac{1}{b} \sum_{\mathbf{x} \in \mathbf{x}_i} \mathcal{H}_{t,i}$ ▷ Average entropy for task t

  Compute $H'_t \leftarrow -\sum_{c=1}^{C_t} \frac{1}{C_t}\ \log \left[\frac{1}{C_t}\right]$ ▷ Max entropy (uniform distribution)

  $\mathcal{B} \leftarrow \mathcal{B} \cup \mathbf{x}_t$ and $D_t \leftarrow D_t \setminus \mathbf{x}_t$

  if $D_t = \emptyset$ **then**

    Reload D_t

  end

  for $i \leftarrow 1$ **to** b **do**

    Compute: $\mathcal{U}_{t,i} \leftarrow \mathcal{H}_{t,i} / H'_t$ ▷ Uncertainty normalized with max entropy

  end

end

Compute $\hat{\mathcal{H}} \leftarrow \max_{i \in \{1,\dots,T\}} [\bar{\mathcal{H}}_t]$ ▷ Entropy of task with highest average entropy

Update $\mathcal{U}_{t,i} \leftarrow \mathcal{U}_{t,i} / \hat{\mathcal{H}}$ ▷ Normalize each sample's uncertainty measure

$\mathcal{B}' \leftarrow$ top_b($\{\mathcal{U}_{t,i} | t \in [1, \dots, T], i \in [1, \dots, b]\}$) ▷ b samples w/ highest uncertainty

Return: With $\mathcal{B}'$, solve eq. 3.1 with gradient descent; updated model f

An advantage of our MT-Uncertainty Sampling approach is its ability to manage task difficulty. This is highlighted in Figure 3.8. In this experiment, we estimated task difficulty using the Evolutionary Data Measures (EDM)[4] proposed by [169]. The task difficulty estimate relies on multiple dataset statistics such as the data size, class diversity, class balance and class interference. Interestingly, estimated task difficulty correlates with the first instance that the selection of a specific task occurs. Supposing that QNLI is an outlier, we notice that peaks in the data occur whenever tasks are first selected by MT Uncertainty sampling. This process follows the following order: 1. MNLI 2. CoLA 3. RTE 4. QQP 5. MRPC 6.SST-2, which is the order from highest task difficulty to lowest task difficulty using EDM. As opposed to Curriculum Learning [170], MT-Uncertainty dynamically prioritizes the most difficult tasks. As also discovered in MTL vision work [171], this type of prioritization on more difficult tasks may explain MT-Uncertainty's improved performance over other task selection methods. In MTL, heuristics to balance tasks during training is typically done by weighting each task's loss differently. We see here how MT-Uncertainty is able to prioritize task difficulty.

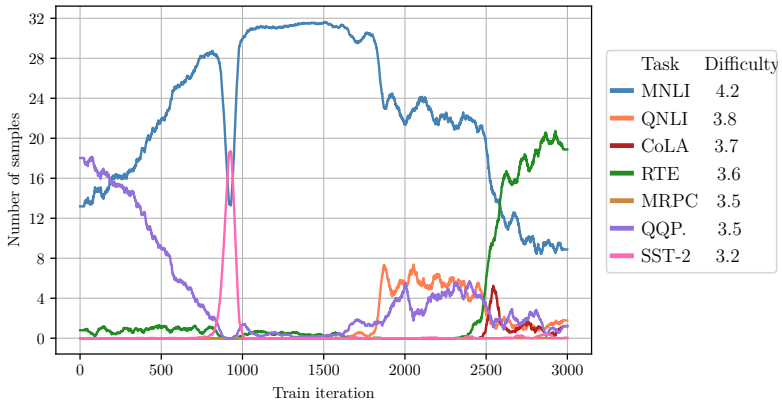


Figure 3.8 Task composition of MT-Uncertainty sampling and estimated task difficulty using EDM: number of training samples per task at each iteration for batch size of 32. The occurrence of first peaks and estimated difficulty follow the same order: From highest to lowest: MNLI > CoLA > RTE > QQP = MRPC > SST-2.

While the EDM difficulty measure is shown to correlate well with model performance, it lacks precision. As reported in [169], the average score achieved on the Yahoo Answers dataset is 69.9% and its difficulty is 4.51. The average score achieved on Yelp Full is 56.8%, 13.1% less than Yahoo Answers and its difficulty is 4.42. The authors mention that "This indicates that the difficulty measure in its current incarnation may be more effective at assigning a class of difficulty to datasets, rather than a regression-like value".

[4]https://github.com/Wluper/edm

3.7.3 Other Related Work

Multi-Tasking in NLP and other fields. MTL weight sharing algorithms such as Mixture-of-Experts (MoE) have found success in NLP [172]. CA-MTL can complement MoE since the Transformers multi-headed attention can be seen as a form of MoE [173]. In Vision, MTL can also improve with optimization [174] or gradient-based approaches [145, 175].

Active Learning, Task Selection and Sampling. [162] examined multi-task active learning for neural semantic role labeling in a low resource setting, using entity recognition as the sole auxiliary task. They used uncertainty sampling for active learning and found that 12% less data could be used compared to passive learning. [161] has examined different active learning techniques for the two task annotation scenario, focusing on named entity recognition and syntactic parse tree annotations. In contrast, here we examine the larger scale data regime, the modularization of a multi-task neural architecture, and the many task ($\gg 2$) setting among other differences. Other than MTAL [161,162], [176] leveraged model uncertainty to balance MTL losses but not to select tasks as is proposed here.

3.7.4 Zero-Shot Results on SciTail and SNLI

Before testing models on domain adaptation in section 3.5.4, we ran zero-shot evaluations on the development set of SciTail and SNLI. Table 3.8 outlines 8-task CA-MTL$_{\text{BERT-BASE}}$'s zero-shot transfer abilities when pretrained on GLUE with our MTL approach. We expand the task embedding layer to accommodate an extra task and explore various embedding initialization. We found that reusing STS-B and MRPC task embeddings worked best for SciTail and SNLI respectively.

3.7.5 More Experimental Details

We used a batch size of 32 and a seed of 12 in all experiments. We used Adam [177] as the optimizer with a learning rate of 2e-5. We applied a learning rate decay with warm up over the first 10% of the training steps. Unless otherwise specified, we used 5 epochs, a seed of 12 and a sequence length of 128. Additional details are outlined in section . Our data prepossessing and linear decoder heads are the same as in [1]. We used the same dropout rate of 0.1 in all layers. To run our experiments, we used either four NVIDIA P100 GPU for base models or four NVIDIA V100 GPU for larger ones. We did not perform parameter search. We do not use ensemble of models or task-specific tricks [1, 4, 154]. All models are either 12 Transformer layers for BASE and 24 Transformer layers for LARGE. Apart from CA-MTL, models trained in multi-task learning (BERT or RoBERTa without adapters) used

Table 3.8 CA-MTL is flexible and extensible to new tasks. However, CA-MTL is sensitive to the new task's embedding. We tested multiple task embeddings that worked best on either SciTail or SNLI by checking performance in a zero shot setting or using 0% of the data.

Initialization of new task embedding layer	SciTail 0% of data	SNLI 0% of data
CoLA's embeddings	43.0	34.0
MNLI's embeddings	24.2	33.0
MRPC's embeddings	34.5	**45.5**
STS-B's embeddings	**46.9**	33.2
SST-2's embeddings	25.8	34.2
QQP's embeddings	31.7	37.3
QNLI's embeddings	32.0	38.0
RTE's embeddings	32.3	40.6
WNLI's embeddings	29.0	30.4
Average	28.7	37.7
Random initialization	46.8	34.0
Xavier initialization	29.8	37.6

random task sampling. For Table 3.1 and Figure 3.7, all BERT-based model have half their layers frozen (untrained) for a fair comparison of ablation results. For the 24-task MTL and CA-MTL models in Tables 3.4 and 3.5, we increased the input sequence length to 256 and used 8 epochs.

3.7.6 The Direct Sum Operator

In section 3.3.1, we used the direct sum operator $\oplus$. This operation allows us to create a block diagonal matrix. The direct sum of a matrix $A \in \mathbb{R}^{n \times m}$ and $B \in \mathbb{R}^{p \times q}$ results in a matrix of size $(m + p) \times (n + q)$, defined as:

$$\mathbf{A} \oplus \mathbf{B} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{1n} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & b_{11} & \cdots & b_{1q} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & b_{p1} & \cdots & b_{pq} \end{bmatrix}$$

3.7.7 Baselines and Other Experimental Results

In this section, we present our baseline results for BERT, RoBERTa, CA-MTL as well as other models. Our single task results (ST) that we ran ourselves surpass other paper's reported scores in Table 3.9. [7] reports random seed median scores for RoBERTa. However, our RoBERTa ST baseline **matches or surpasses** the original paper's scores **4 out 7 times** on the development set when scores are comparable (QQP F1 and STS-B spearman are not reported).

Table 3.9 F1 scores are reported for QQP/MRPC, Spearman's correlation for STS-B, accuracy on the matched/mismatch sets for MNLI, Matthew's correlation for CoLA and accuracy for other tasks. ST=Single Task, MTL=Multitask. *QNLI v1 (we report v2) **F1 score or Spearman's correlation is not reported. ***Unknown random seeds. Results from: [1] [2] [2] [4] [3] [6] [4] [7].

Method	Total params	Trained params/task	GLUE CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	Avg
Base Models — Dev set Results											
PALs+Anneal Samp.[1]	1.13×	12.5%	–	–	–	–	–	–	–	–	81.70
8-task CA-MTL$_{\text{BERT-BASE}}$ (ours)	1.12×	5.6%	60.9	82.7/83.1	88.9	90.7	90.3	79.1	91.9	88.8	84.03
BERT LARGE Models — Dev set Results											
ST BERT-LARGE[2]	9×	100%	60.5	86.7/85.9	89.3	92.7*	89.3	70.1	94.9	86.5	84.0
ST BERT-LARGE[3]	9×	100%	62.1	86.2/86.2	92.3	89.4	88.5	70.0	92.5	90.1	84.1
ST BERT-LARGE (ours)	9×	100%	63.6	86.5/86.0	91.4	91.0	88.5	70.2	94.7	88.2	84.5
24-task CA-MTL$_{\text{BERT-LARGE}}$ (ours)	1.12×	5.6%	63.8	86.3/86.0	92.9	93.4	88.1	84.5	94.5	90.3	86.6
RoBERTa LARGE Models — Dev set Results											
RoBERTa-LARGE[4] (Median 5 runs)***	9×	100%	68.0	90.2	90.9	94.7	**	86.6	96.4	**	–
ST RoBERTa-LARGE (ours)	9×	100%	68.3	89.2/88.9	92.6	94.8	84.6	87.0	96.4	91.7	88.2
24-task CA-MTL$_{\text{RoBERTa-LARGE}}$ (ours)	1.12×	5.6%	69.7	89.4/89.3	93.9	94.9	88.8	91.0	96.2	91.0	89.4

3.7.8 Some Results on layer Freezing and with Full Block Attention.

All experiments in this section were run for only 5 epochs, exclusively on the GLUE dataset for the large BERT-based 8-task CA-MTL model. Results in Table 3.10 reveal that as we freeze more layers, performance tends to decrease. However, since we wanted to preserve as much pretrained knowledge as possible, we chose to keep at least 50% of layers frozen. While this has slightly lowered our performance on 9 GLUE tasks, we believe that keeping as much of the original pretrained weights is beneficial when increasing the total number of tasks in MTL to 24 or more tasks. However, we did not explore this hypothesis more.

Table 3.10 **8-task CA-MTL$_{\text{BERT-LARGE}}$** (see section 3.5.3) for various layer freezing configurations. F1 scores are reported for QQP/MRPC, Spearman's correlation for STS-B, accuracy on the matched/mismatch sets for MNLI, Matthew's correlation for CoLA and accuracy for other tasks. FBA = Full Block Attention

Method	% frozen layers	# tasks g.e ST	GLUE								
			CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	Avg
LARGE Models — Dev set Results											
ST BERT-LARGE (ours)	0%	—	63.6	86.5/86.0	91.4	91.0	88.5	70.2	93.1	88.2	84.3
CA-MTL	0%	7	60.2	86.2/86.0	92.0	91.5	88.7	76.3	93.3	89.5	84.9
CA-MTL	25%	6	63.7	86.1/85.8	89.1	91.2	88.6	79.7	92.9	88.5	85.1
CA-MTL	50%	3	63.2	85.5/85.5	91.8	90.9	88.3	81.4	93.0	90.1	85.5
CA-MTL FBA	50%	0	60.2	81.7/81.1	88.0	85.8	85.7	78.7	88.6	87.1	81.8

3.7.9 Dataset Description

The datasets that were used for the domain adaptation experiments were SciTail[5] [6] and SNLI[7]. We *jointly* trained a CA-MTL$_{\text{RoBERTa-LARGE}}$ model on 9 GLUE tasks, 8 Super-GLUE[8] tasks, 6 MRQA[9] tasks, and on WNUT2017[10] [166].

Table 3.11 GLUE [8] dataset description. References: [1] [9], [2] [10], [3] [11], [4] [12], [5] [13], [6] [8], [7] [14]

Acronym	**Corpus**	**|Train|**	**Task**	**Domain**
CoLA[1]	Corpus of Linguistic Acceptability	8.5K	acceptability	miscellaneous
SST-2[2]	Stanford Sentiment Treebank	67K	sentiment detection	movie reviews
MRPC[3]	Microsoft Research Paraphrase Corpus	3.7K	paraphrase detection	news
STS-B[4]	Semantic Textual Similarity Benchmark	7K	textual similarity	miscellaneous
QQP	Quora Question Pairs	364K	paraphrase detection	online QA
MNLI[5]	Multi-Genre NLI	393K	inference	miscellaneous
RTE[6]	Recognition Textual Entailment	2.5K	inference/entailment	news, Wikipedia
WNLI[7]	Winograd NLI	634	coreference	fiction books

All GLUE tasks are binary classification, except STS-B (regression) and MNLI (three classes). We used the same GLUE data preprocessing as in [1].

SuperGLUE has a more diverse task format than GLUE, which is mostly limited to sentence and sentence-pair classification. We follow the same preprocessing procedure as in [15]. All tasks are binary classification tasks, except CB (three classes). Also, WiC and WSC are span based classification tasks. We used the same modified MRQA dataset and preprocessing steps

[5]https://allenai.org/data/scitail
[6]SciTail Leaderbard: https://leaderboard.allenai.org/scitail/submissions/public
[7]https://nlp.stanford.edu/projects/snli/
[8]https://super.gluebenchmark.com/tasks
[9]https://github.com/mrqa/MRQA-Shared-Task-2019
[10]https://github.com/leondz/emerging_entities_17

Table 3.12 Super-GLUE [15] dataset description. References: [1] [16], [2] [17], [3] [18], [4] [19], [5] [20], [6] [15], [7] [21], [8] [14]

Acronym	Corpus	|Train|	Task	Domain
BoolQ[1]	Boolean Questions	9.4K	acceptability	Google queries, Wikipedia
CB[2]	CommitmentBank	250	sentiment detection	miscellaneous
COPA[3]	Choice of Plausible Alternatives	400	paraphrase detection	blogs, encyclopedia
MultiRC[4]	Multi-Sentence Reading Comprehension	5.1K	textual similarity	miscellaneous
ReCoRD[5]	Reading Comprehension and Commonsense Reasoning	101K	paraphrase detection	news
RTE[6]	Recognition Textual Entailment	2.5K	inference	news, Wikipedia
WiC[7]	Word-in-Context	6K	word sense disambiguation	WordNet, VerbNet
WSC[8]	Winograd Schema Challenge	554	coreference resolution	fiction books

Table 3.13 MRQA [22] dataset description. References: [1] [23], [2] [24], [3] [25], [4] [26], [5] [27], [6] [28]

Acronym	Corpus	|Train|	Task	Domain
SQuAD[1]	Stanford QA Dataset	86.6K	crowdsourced questions	Wikipedia
NewsQA[2]	NewsQA	74.2K	crowdsourced questions	news
TriviaQA[3]	TriviaQA	61.7K	trivia QA	web snippets
SearchQA[4]	SearchQA	117.4K	Jeopardy QA	web snippets
HotpotQA[5]	HotpotQA	72.9K	crowdsourced questions	Wikipedia
Natural Questions[6]	Natural Questions	104.7K	search logs	Wikipedia

that were used in [178]. All MRQA tasks are span prediction tasks which seeks to identify start and end tokens of an answer span in the input text.

Table 3.14 SNLI [29] and SciTail [30] datasets description.

Acronym	Corpus	|Train|	Task	Domain
SNLI[1]	Stanford Natural Language Inference	550.2k	inference	human-written English sentence pairs
SciTail[2]	Science and Entailment	23.5K	entailment	Science question answering

SNLI is a natural inference task where we predict three classes. Examples of three target labels are: Entailment, Contradiction, and Neutral (irrelevant). SciTail is a textual entailment dataset. The hypotheses in SciTail are created from multiple-choice science exams and the answer candidates (premise) are extracted from the web using information retrieval tools. SciTail is a binary true/false classification tasks that seeks to predict whether the premise entails the hypothesis. The two datasets are used only for domain adaptation in this study (see section 3.7.4 for the details of our approach).

CHAPTER 4 GENERAL DISCUSSION

In this thesis, we achieved all our objectives with some limitations presented in Section 5.2.

The main objective of this thesis was to explore the possibility of using MTL alone to outperform STL fine-tuning on various NLP tasks. The results in Section 3.5.3 shows that we were able to achieve this objective. First, we compared our approach to multiple single task fine-tuned models and recent BERT-based adapters against the GLUE benchmark. In Table 3.2, we show that CA-MTL outperforms all other models on both the development and test sets. Also, our BERT **base** variant outperforms a single task fine-tuned BERT **large** model. Second, we jointly trained a ROBERTA variant of CA-MTL on 24 NLP tasks from different domains to validate the scalability of our approach. We compared our results to T5 baseline model for classification tasks (Super-GLUE) and SpanBERT for question answering tasks (MRQA). Our model outperforms T5 base model by 6 percentage points on the Super-GLUE development set and performs competitively on the MRQA development set compared to SpanBERT. We think that our sampling algorithm affected our perfomance on question answering tasks (more detail in Section 5.2).

Our approach is parameter efficient. The performance of our ROBERTA base variant is 2% higher than a fine-tuned BERT large model that has almost double the parameters count. Also, it is important to note that a fine-tuning approach requires n-times the total parameter count, where n is the number of tasks. For example, The GLUE benchmark contains 9 tasks, a fine-tuning approach will require 9x the parameter count, while CA-MTL only requires 1.12x since our approach reuse parameters. In the presence of multiple tasks, the efficiency of our approach shows very useful. Especially in an environment where access to compute resources is limited (like an academic environment). We were able to run many of our experiments using the small NVIDIA P100 GPU.

Our approach is data efficient. Using our sampling technique, CA-MTL uses less training data to achieve better performance. MT-Uncertainty reduces training time. It manages varying task difficulties by choosing most difficult tasks first using an uncertainty measure. This task scheduling mechanism limited task interference. Also, it avoids catastrophic forgetting for low resource tasks, since a task is sampled when the model becomes uncertain even if dataset is entirely seen within an epoch. Finally, MT-Uncertainty converges fast.

We believe freezing the first layers of our model, limits the pretraining knowledge loss. Our domain adaptation experiment where we limited the training data showed that low resource regimes benefits from this technique. However, this benefit is not fully understood. We think

that deep analysis of the leaned representations is required, as discussed in Section 5.3.

CA-MTL adapts well to new tasks. We performed domain adaptation on SNLI and SciTail datasets. Our model achieved SOTA results in both datasets. Also, using small portions of data, we proved that CA-MTL adapts fast in low resource regimes (0.1% and 1% of the data) compared to MT-DNN_SMART (the previous SOTA on SciTail dataset). This makes our model very useful in domain adaptation cases where labeled data is scarce. However, our task embeddings initialization technique requires more refinement as discussed in Section 5.2.

Finally, CA-MTL is adaptable to any existing Transformer architecture. In our experiments we were able interchange between BERT and ROBERTA without any architectural change.

CHAPTER 5 CONCLUSION

5.1 Summary of Works

Our approach (CA-MTL) outperformed fine-tuned STL models, only adding 5.6% more parameters and only using 64.6% of the training data. We believe, this result is die to by the cross knowledge between tasks, enhanced by our adaptive modules and the pre-training knowledge retention achieved by freezing the first layer of the model. Also, our MT-Uncertainty sampling technique showed very effective in handling low resource data, using fewer data and managing varying task difficulties during training. Our task conditioning modules showed very extensible in a large scale MTL setup (using 26 NLP tasks) compared to a standard MTL setup.

CA-MTL achieved SOTA results on multiple test and development sets showing the efficacy of our approach.

5.2 Limitations

While we achieved SOTA results on SNLI and SciTail datasets using domain adaptation, the task embeddings initialization isn't scalable. We initialized the task embeddings for the adapted tasks by choosing the best performing one from a previous run MTL experiment or random embeddings.

Our MT-Uncertainty sampling technique showed very effective. However, it adds a "heavy" computation cost, increasing the training time by 1.5x. Moreover, this technique only works for classification tasks. The approach selects regression tasks using proxy tasks that were deemed related. Further, we noticed a "strange" behavior where some question answering tasks aren't sampled for a long period of time. This might explain the lower performance on some question answering tasks.

5.3 Future Research

Although we can explain some reasons why CA-MTL performs well. A detailed analysis of the representation learned by our model could provide more insight on the weight sharing between tasks and knowledge retention. For domain adaptation, future work should focus on finding a better task embeddings initialization scheme. We also think that our sampling strategy (MT-Uncertainty) needs more improvement, mainly reducing the added computational cost

and better handling regression and question answering tasks.

Our study showed that the more tasks we add, the better the overall performance. While it would require further computational resources, the following questions are interesting venues for future work: Is there a limit on how much tasks we can add? Will CA-MTL perform better with a sequence length of 512 instead of 256 used in our best model? Will CA-MTL perform even better with a larger model?

REFERENCES

[1] J. Devlin *et al.*, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[2] A. C. Stickland *et al.*, "BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning," ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 5986–5995. [Online]. Available: http://proceedings.mlr.press/v97/stickland19a.html

[3] N. Houlsby *et al.*, "Parameter-efficient transfer learning for NLP," *CoRR*, vol. abs/1902.00751, 2019. [Online]. Available: http://arxiv.org/abs/1902.00751

[4] X. Liu *et al.*, "Multi-task deep neural networks for natural language understanding," *CoRR*, vol. abs/1901.11504, 2019. [Online]. Available: http://arxiv.org/abs/1901.11504

[5] H. Jiang *et al.*, "SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 2177–2190. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.197

[6] J. Phang, T. Févry, and S. R. Bowman, "Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks," *CoRR*, vol. abs/1811.01088, 2018. [Online]. Available: http://arxiv.org/abs/1811.01088

[7] Y. Liu *et al.*, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[8] A. Wang *et al.*, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 353–355. [Online]. Available: https://www.aclweb.org/anthology/W18-5446

[9] A. Warstadt, A. Singh, and S. R. Bowman, "Neural network acceptability judgments," *CoRR*, vol. abs/1805.12471, 2018. [Online]. Available: http://arxiv.org/abs/1805.12471

[10] R. Socher *et al.*, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642. [Online]. Available: https://www.aclweb.org/anthology/D13-1170

[11] W. B. Dolan and C. Brockett, "Automatically constructing a corpus of sentential paraphrases," in *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005. [Online]. Available: https://www.aclweb.org/anthology/I05-5002

[12] D. Cer *et al.*, "SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 1–14. [Online]. Available: https://www.aclweb.org/anthology/S17-2001

[13] A. Williams, N. Nangia, and S. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 1112–1122. [Online]. Available: https://www.aclweb.org/anthology/N18-1101

[14] H. J. Levesque, "The winograd schema challenge." in *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*. AAAI, 2011. [Online]. Available: http://dblp.uni-trier.de/db/conf/aaaiss/aaaiss2011-6.html#Levesque11

[15] A. Wang *et al.*, "Superglue: A stickier benchmark for general-purpose language understanding systems," *CoRR*, vol. abs/1905.00537, 2019. [Online]. Available: http://arxiv.org/abs/1905.00537

[16] C. Clark *et al.*, "BoolQ: Exploring the surprising difficulty of natural yes/no questions," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 2924–2936. [Online]. Available: https://www.aclweb.org/anthology/N19-1300

[17] M.-C. de Marneffe, M. Simons, and J. Tonhauser, "The commitmentbank: Investigating projection in naturally occurring discourse," 2019.

[18] A. Gordon, Z. Kozareva, and M. Roemmele, "SemEval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning," in **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*. Montréal, Canada: Association for Computational Linguistics, 7-8 Jun. 2012, pp. 394–398. [Online]. Available: https://www.aclweb.org/anthology/S12-1052

[19] D. Khashabi *et al.*, "Looking beyond the surface: A challenge set for reading comprehension over multiple sentences," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 252–262. [Online]. Available: https://www.aclweb.org/anthology/N18-1023

[20] S. Zhang *et al.*, "Record: Bridging the gap between human and machine commonsense reading comprehension," *CoRR*, vol. abs/1810.12885, 2018. [Online]. Available: http://arxiv.org/abs/1810.12885

[21] A. Poliak *et al.*, "Collecting diverse natural language inference problems for sentence representation evaluation," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 67–81. [Online]. Available: https://www.aclweb.org/anthology/D18-1007

[22] A. Fisch *et al.*, "Mrqa 2019 shared task: Evaluating generalization in reading comprehension," 2019.

[23] P. Rajpurkar *et al.*, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392. [Online]. Available: https://www.aclweb.org/anthology/D16-1264

[24] A. Trischler *et al.*, "NewsQA: A machine comprehension dataset," in *Proceedings of the 2nd Workshop on Representation Learning for NLP*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 191–200. [Online]. Available: https://www.aclweb.org/anthology/W17-2623

[25] M. Joshi *et al.*, "TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension," in *Proceedings of the 55th Annual Meeting of the*

Association for Computational Linguistics (Volume 1: Long Papers). Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 1601–1611. [Online]. Available: https://www.aclweb.org/anthology/P17-1147

[26] M. Dunn *et al.*, "Searchqa: A new q&a dataset augmented with context from a search engine," *CoRR*, vol. abs/1704.05179, 2017. [Online]. Available: http://arxiv.org/abs/1704.05179

[27] Z. Yang *et al.*, "HotpotQA: A dataset for diverse, explainable multi-hop question answering," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 2369–2380. [Online]. Available: https://www.aclweb.org/anthology/D18-1259

[28] T. Kwiatkowski *et al.*, "Natural questions: a benchmark for question answering research," *Transactions of the Association of Computational Linguistics*, 2019.

[29] S. R. Bowman *et al.*, "A large annotated corpus for learning natural language inference," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

[30] T. Khot, A. Sabharwal, and P. Clark, "Scitail: A textual entailment dataset from science question answering," in *AAAI*, 2018.

[31] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[32] S. Ruder, "Neural transfer learning for natural language processing," Ph.D. dissertation, National University of Ireland, Galway, 2019.

[33] ——, "An overview of multi-task learning in deep neural networks," *CoRR*, vol. abs/1706.05098, 2017. [Online]. Available: http://arxiv.org/abs/1706.05098

[34] H. Martínez Alonso and B. Plank, "When is multitask learning effective? Semantic sequence prediction under varying data conditions," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 44–53. [Online]. Available: https://www.aclweb.org/anthology/E17-1005

[35] L. Duong *et al.*, "Low Resource Dependency Parsing: Cross-lingual Parameter Sharing in a Neural Network Parser," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 845–850. [Online]. Available: https://www.aclweb.org/anthology/P15-2139

[36] A. Vaswani *et al.*, "Attention Is All You Need," *arXiv:1706.03762 [cs]*, Dec. 2017, arXiv: 1706.03762. [Online]. Available: http://arxiv.org/abs/1706.03762

[37] N. Houlsby *et al.*, "Parameter-Efficient Transfer Learning for NLP," *arXiv:1902.00751 [cs, stat]*, Jun. 2019, arXiv: 1902.00751. [Online]. Available: http://arxiv.org/abs/1902.00751

[38] A. C. Stickland and I. Murray, "BERT and PALs: Projected Attention Layers for Efficient Adaptation in Multi-Task Learning," *arXiv:1902.02671 [cs, stat]*, May 2019, arXiv: 1902.02671. [Online]. Available: http://arxiv.org/abs/1902.02671

[39] J. Glover and C. Hokamp, "Task selection policies for multitask learning," *CoRR*, 2019. [Online]. Available: http://arxiv.org/abs/1907.06214

[40] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 328–339. [Online]. Available: https://www.aclweb.org/anthology/P18-1031

[41] M. Peters *et al.*, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 2227–2237. [Online]. Available: https://www.aclweb.org/anthology/N18-1202

[42] A. Williams, N. Nangia, and S. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 1112–1122. [Online]. Available: https://www.aclweb.org/anthology/N18-1101

[43] M. Peters *et al.*, "Semi-supervised sequence tagging with bidirectional language models," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 1756–1765. [Online]. Available: https://www.aclweb.org/anthology/P17-1161

[44] V. Joshi, M. Peters, and M. Hopkins, "Extending a parser to distant domains using a few dozen partially annotated examples," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 1190–1199. [Online]. Available: https://www.aclweb.org/anthology/P18-1110

[45] A. Vaswani *et al.*, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[46] A. Radford *et al.*, "Language models are unsupervised multitask learners (gpt)," 2019.

[47] T. Pires, E. Schlinger, and D. Garrette, "How multilingual is multilingual bert?" *CoRR*, vol. abs/1906.01502, 2019. [Online]. Available: http://arxiv.org/abs/1906.01502

[48] Z. Yang *et al.*, "Xlnet: Generalized autoregressive pretraining for language understanding," *CoRR*, vol. abs/1906.08237, 2019. [Online]. Available: http://arxiv.org/abs/1906.08237

[49] Z. Lan *et al.*, "Albert: A lite bert for self-supervised learning of language representations," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=H1eA7AEtvS

[50] Y. Liu *et al.*, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[51] J. Howard and S. Ruder, "Universal Language Model Fine-tuning for Text Classification," *arXiv:1801.06146 [cs, stat]*, May 2018, arXiv: 1801.06146. [Online]. Available: http://arxiv.org/abs/1801.06146

[52] K. Clark *et al.*, "What does BERT look at? an analysis of bert's attention," *CoRR*, vol. abs/1906.04341, 2019. [Online]. Available: http://arxiv.org/abs/1906.04341

[53] I. Tenney, D. Das, and E. Pavlick, "BERT rediscovers the classical NLP pipeline," *CoRR*, vol. abs/1905.05950, 2019. [Online]. Available: http://arxiv.org/abs/1905.05950

[54] A. Coenen *et al.*, "Visualizing and measuring the geometry of BERT," *CoRR*, vol. abs/1906.02715, 2019. [Online]. Available: http://arxiv.org/abs/1906.02715

[55] X. Liu *et al.*, "Multi-task deep neural networks for natural language understanding," *CoRR*, vol. abs/1901.11504, 2019. [Online]. Available: http://arxiv.org/abs/1901.11504

[56] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 160–167. [Online]. Available: https://doi.org/10.1145/1390156.1390177

[57] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015. [Online]. Available: http://arxiv.org/abs/1504.08083

[58] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: an overview," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 8599–8603.

[59] B. Ramsundar *et al.*, "Massively multitask networks for drug discovery," *arXiv preprint arXiv:1502.02072*, 2015.

[60] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, Jul. 1997. [Online]. Available: https://doi.org/10.1023/A:1007379606734

[61] ——, "Multitask learning: A knowledge-based source of inductive bias," in *Proceedings of the Tenth International Conference on Machine Learning*. Morgan Kaufmann, 1993, pp. 41–48.

[62] J. Pilault, A. E. hattami, and C. Pal, "Conditionally adaptive multi-task learning: Improving transfer learning in {nlp} using fewer parameters & less data," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=de11dbHzAMF

[63] S. Ben-David *et al.*, "A theory of learning from different domains," *Mach. Learn.*, vol. 79, no. 1–2, p. 151–175, May 2010. [Online]. Available: https://doi.org/10.1007/s10994-009-5152-4

[64] Y. Pruksachatkun *et al.*, "Intermediate-task transfer learning with pretrained models for natural language understanding: When and why does it work?" 2020.

[65] R. Raina *et al.*, "Self-taught learning: Transfer learning from unlabeled data," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 759–766. [Online]. Available: https://doi.org/10.1145/1273496.1273592

[66] B. Zadrozny, "Learning and evaluating classifiers under sample selection bias." New York, NY, USA: Association for Computing Machinery, 2004. [Online]. Available: https://doi.org/10.1145/1015330.1015425

[67] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of Statistical Planning and Inference*, vol. 90, pp. 227–244, 2000.

[68] H. Daumé and D. Marcu, "Domain adaptation for statistical classifiers," *J. Artif. Int. Res.*, vol. 26, no. 1, p. 101–126, May 2006.

[69] A. Arnold, R. Nallapati, and W. W. Cohen, "A comparative study of methods for transductive transfer learning," in *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*, 2007, pp. 77–82.

[70] Z. Wang, Y. Song, and C. Zhang, "Transferred dimensionality reduction," in *Machine Learning and Knowledge Discovery in Databases*, W. Daelemans, B. Goethals, and K. Morik, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 550–565.

[71] A. Conneau *et al.*, "Unsupervised cross-lingual representation learning at scale," 2020.

[72] W. Dai *et al.*, "Boosting for transfer learning," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 193–200. [Online]. Available: https://doi.org/10.1145/1273496.1273521

[73] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, p. 119–139, Aug. 1997. [Online]. Available: https://doi.org/10.1006/jcss.1997.1504

[74] M. H. Kalos and P. A. Whitlock, *Monte Carlo Methods. Vol. 1: Basics*. USA: Wiley-Interscience, 1986.

[75] J. Meng, H. Lin, and Y. Li, "Knowledge transfer based on feature representation mapping for text classification," *Expert Systems with Applications*, vol. 38, no. 8,

pp. 10 562 – 10 567, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417411002910

[76] L. Duan, I. W. Tsang, and D. Xu, "Domain transfer multiple kernel learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 465–479, 2012.

[77] H. Lee *et al.*, "Exponential family sparse coding with applications to self-taught learning," in *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, ser. IJCAI'09. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009, p. 1113–1119.

[78] L. Mihalkova, T. Huynh, and R. J. Mooney, "Mapping and revising markov logic networks for transfer learning," in *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 1*, ser. AAAI'07. AAAI Press, 2007, p. 608–614.

[79] B. L. Richards and R. Mooney, "Automated refinement of first-order horn-clause domain theories," *Machine Learning*, vol. 19, pp. 95–131, 2004.

[80] M. T. Rosenstein *et al.*, "To transfer or not to transfer," in *In NIPS'05 Workshop, Inductive Transfer: 10 Years Later*, 2005.

[81] Z. Wang *et al.*, "Characterizing and avoiding negative transfer," 2019.

[82] S. Ben-David and R. Borbely, "Exploiting task relatedness for multiple task learning," *Lecture Notes in Computer Science*, vol. 2777, 08 2002.

[83] B. Bakker and T. Heskes, "Task clustering and gating for bayesian multitask learning," *J. Mach. Learn. Res.*, vol. 4, no. null, p. 83–99, Dec. 2003. [Online]. Available: https://doi.org/10.1162/153244304322765658

[84] R. Caruana, "Multitask Learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, Jul. 1997. [Online]. Available: https://doi.org/10.1023/A:1007379606734

[85] Y. Xue *et al.*, "Multi-Task Learning for Classification with Dirichlet Process Priors," *The Journal of Machine Learning Research*, vol. 8, pp. 35–63, Dec. 2007.

[86] O. Sener and V. Koltun, "Multi-Task Learning as Multi-Objective Optimization," *arXiv:1810.04650 [cs, stat]*, Jan. 2019, arXiv: 1810.04650. [Online]. Available: http://arxiv.org/abs/1810.04650

[87] J. Baxter, "A Bayesian/Information Theoretic Model of Learning to Learn via Multiple Task Sampling," *Machine Learning*, vol. 28, no. 1, pp. 7–39, Jul. 1997. [Online]. Available: https://doi.org/10.1023/A:1007327622663

[88] A. Wang *et al.*, "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 353–355. [Online]. Available: https://www.aclweb.org/anthology/W18-5446

[89] S. R. Bowman *et al.*, "A large annotated corpus for learning natural language inference," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 632–642. [Online]. Available: https://www.aclweb.org/anthology/D15-1075

[90] T. Khot, A. Sabharwal, and P. Clark, "SciTaiL: A Textual Entailment Dataset from Science Question Answering," in *AAAI*, 2018.

[91] X. Liu *et al.*, "Stochastic Answer Networks for Machine Reading Comprehension," *arXiv:1712.03556 [cs]*, May 2018, arXiv: 1712.03556. [Online]. Available: http://arxiv.org/abs/1712.03556

[92] J. Bjerva, B. Plank, and J. Bos, "Semantic Tagging with Deep Residual Networks," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, Dec. 2016, pp. 3531–3541. [Online]. Available: https://www.aclweb.org/anthology/C16-1333

[93] Y. Yang and T. M. Hospedales, "Trace Norm Regularised Deep Multi-Task Learning," *arXiv:1606.04038 [cs]*, Feb. 2017, arXiv: 1606.04038. [Online]. Available: http://arxiv.org/abs/1606.04038

[94] M. Yuan *et al.*, "Dimension reduction and coefficient estimation in multivariate linear regression," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 69, no. 3, pp. 329–346, 2007, _eprint: https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9868.2007.00591.x. [Online]. Available: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2007.00591.x

[95] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization," *SIAM Review*, vol. 52, no. 3, pp. 471–501, Jan. 2010, arXiv: 0706.4138. [Online]. Available: http://arxiv.org/abs/0706.4138

[96] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, Sep. 1966. [Online]. Available: https://doi.org/10.1007/BF02289464

[97] L. Eldén and B. Savas, "A Newton–Grassmann Method for Computing the Best Multilinear Rank-$(r_1,$ $r_2,$ $r_3)$ Approximation of a Tensor," *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 2, pp. 248–271, Jan. 2009, publisher: Society for Industrial and Applied Mathematics. [Online]. Available: https://epubs.siam.org/doi/10.1137/070688316

[98] I. Oseledets, "Tensor-Train Decomposition," *SIAM J. Scientific Computing*, vol. 33, pp. 2295–2317, Jan. 2011.

[99] T. Yu *et al.*, "Gradient Surgery for Multi-Task Learning," *arXiv:2001.06782 [cs, stat]*, Oct. 2020, arXiv: 2001.06782. [Online]. Available: http://arxiv.org/abs/2001.06782

[100] D. Angluin, "Queries and Concept Learning," *Machine Learning*, vol. 2, no. 4, pp. 319–342, Apr. 1988. [Online]. Available: https://doi.org/10.1023/A:1022821128753

[101] R. D. King *et al.*, "The Automation of Science," *Science*, vol. 324, no. 5923, pp. 85–89, Apr. 2009, publisher: American Association for the Advancement of Science Section: Report. [Online]. Available: https://science.sciencemag.org/content/324/5923/85

[102] D. D. Lewis and W. A. Gale, "A Sequential Algorithm for Training Text Classifiers," *arXiv:cmp-lg/9407020*, Jul. 1994, arXiv: cmp-lg/9407020. [Online]. Available: http://arxiv.org/abs/cmp-lg/9407020

[103] L. E. Atlas, D. A. Cohn, and R. E. Ladner, "Training Connectionist Networks with Queries and Selective Sampling," in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, Ed. Morgan-Kaufmann, 1990, pp. 566–573. [Online]. Available: http://papers.nips.cc/paper/261-training-connectionist-networks-with-queries-and-selective-sampling.pdf

[104] I. Dagan and S. P. Engelson, "Committee-Based Sampling For Training Probabilistic Classifiers," in *Machine Learning Proceedings 1995*, A. Prieditis and S. Russell, Eds.

San Francisco (CA): Morgan Kaufmann, Jan. 1995, pp. 150–157. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B978155860377650027X

[105] T. M. Mitchell, "Generalization as search," *Artificial Intelligence*, vol. 18, no. 2, pp. 203–226, Mar. 1982. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0004370282900406

[106] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ser. COLT '92. New York, NY, USA: Association for Computing Machinery, 1992, p. 287–294. [Online]. Available: https://doi.org/10.1145/130385.130417

[107] S. Dasgupta, D. J. Hsu, and C. Monteleoni, "A general agnostic active learning algorithm," in *Advances in Neural Information Processing Systems 20*, J. C. Platt *et al.*, Eds. Curran Associates, Inc., 2008, pp. 353–360. [Online]. Available: http://papers.nips.cc/paper/3325-a-general-agnostic-active-learning-algorithm.pdf

[108] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *Proceedings of the fifth annual workshop on Computational learning theory*, ser. COLT '92. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, Jul. 1992, pp. 287–294. [Online]. Available: https://doi.org/10.1145/130385.130417

[109] B. Settles, M. Craven, and S. Ray, "Multiple-Instance Active Learning," in *Advances in Neural Information Processing Systems 20*, J. C. Platt *et al.*, Eds. Curran Associates, Inc., 2008, pp. 1289–1296. [Online]. Available: http://papers.nips.cc/paper/3252-multiple-instance-active-learning.pdf

[110] N. Roy and A. Mccallum, "Toward Optimal Active Learning through Sampling Estimation of Error Reduction," in *In Proc. 18th International Conf. on Machine Learning*. Morgan Kaufmann, 2001, pp. 441–448.

[111] B. Settles and M. Craven, "An analysis of active learning strategies for sequence labeling tasks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '08. Honolulu, Hawaii: Association for Computational Linguistics, Oct. 2008, pp. 1070–1079.

[112] P. E. Utgoff, "Improved training via incremental learning," in *Proceedings of the sixth international workshop on Machine learning*. Ithaca, New York, USA: Morgan Kaufmann Publishers Inc., Dec. 1989, pp. 362–365.

[113] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, Jul. 1948, conference Name: The Bell System Technical Journal.

[114] R. Hwa, "Sample Selection for Statistical Parsing," *Computational Linguistics*, vol. 30, no. 3, pp. 253–276, 2004. [Online]. Available: https://www.aclweb.org/anthology/J04-3001

[115] T. Scheffer, C. Decomain, and S. Wrobel, "Active Hidden Markov Models for Information Extraction," in *Advances in Intelligent Data Analysis*, ser. Lecture Notes in Computer Science, F. Hoffmann *et al.*, Eds. Berlin, Heidelberg: Springer, 2001, pp. 309–318.

[116] C. Körner and S. Wrobel, "Multi-class Ensemble-Based Active Learning," in *Machine Learning: ECML 2006*, ser. Lecture Notes in Computer Science, J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, Eds. Berlin, Heidelberg: Springer, 2006, pp. 687–694.

[117] R. Reichart *et al.*, "Multi-Task Active Learning for Linguistic Annotations," in *Proceedings of ACL-08: HLT*. Columbus, Ohio: Association for Computational Linguistics, Jun. 2008, pp. 861–869. [Online]. Available: https://www.aclweb.org/anthology/P08-1098

[118] F. Ikhwantri *et al.*, "Multi-Task Active Learning for Neural Semantic Role Labeling on Low Resource Conversational Corpus," in *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*. Melbourne: Association for Computational Linguistics, Jul. 2018, pp. 43–50. [Online]. Available: https://www.aclweb.org/anthology/W18-3406

[119] J. Lafferty, A. Mccallum, and F. Pereira, *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*, 2001.

[120] D. Kottke *et al.*, "Limitations of Assessing Active Learning Performance at Runtime," Jan. 2019. [Online]. Available: https://arxiv.org/abs/1901.10338v1

[121] T. B. Brown *et al.*, "Language Models are Few-Shot Learners," *arXiv:2005.14165 [cs]*, Jul. 2020, arXiv: 2005.14165. [Online]. Available: http://arxiv.org/abs/2005.14165

[122] J. Devlin *et al.*, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv:1810.04805 [cs]*, May 2019, arXiv: 1810.04805. [Online]. Available: http://arxiv.org/abs/1810.04805

[123] N. Carion *et al.*, "End-to-End Object Detection with Transformers," *arXiv:2005.12872 [cs]*, May 2020, arXiv: 2005.12872. [Online]. Available: http://arxiv.org/abs/2005.12872

[124] M. Dehghani *et al.*, "Universal Transformers," *arXiv:1807.03819 [cs, stat]*, Mar. 2019, arXiv: 1807.03819. [Online]. Available: http://arxiv.org/abs/1807.03819

[125] K. Ahmed, N. S. Keskar, and R. Socher, "Weighted Transformer Network for Machine Translation," *arXiv:1711.02132 [cs]*, Nov. 2017, arXiv: 1711.02132. [Online]. Available: http://arxiv.org/abs/1711.02132

[126] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The Efficient Transformer," *arXiv:2001.04451 [cs, stat]*, Feb. 2020, arXiv: 2001.04451. [Online]. Available: http://arxiv.org/abs/2001.04451

[127] Y. Tay *et al.*, "Sparse Sinkhorn Attention," *arXiv:2002.11296 [cs]*, Feb. 2020, arXiv: 2002.11296. [Online]. Available: http://arxiv.org/abs/2002.11296

[128] J. W. Rae *et al.*, "Compressive Transformers for Long-Range Sequence Modelling," *arXiv:1911.05507 [cs, stat]*, Nov. 2019, arXiv: 1911.05507. [Online]. Available: http://arxiv.org/abs/1911.05507

[129] K. Clark *et al.*, "What Does BERT Look At? An Analysis of BERT's Attention," *arXiv:1906.04341 [cs]*, Jun. 2019, arXiv: 1906.04341. [Online]. Available: http://arxiv.org/abs/1906.04341

[130] I. Tenney *et al.*, "What do you learn from context? Probing for sentence structure in contextualized word representations," *arXiv:1905.06316 [cs]*, May 2019, arXiv: 1905.06316. [Online]. Available: http://arxiv.org/abs/1905.06316

[131] N. F. Liu *et al.*, "Linguistic Knowledge and Transferability of Contextual Representations," *arXiv:1903.08855 [cs]*, Apr. 2019, arXiv: 1903.08855. [Online]. Available: http://arxiv.org/abs/1903.08855

[132] H. de Vries *et al.*, "Modulating early visual processing by language," *arXiv:1707.00683 [cs]*, Dec. 2017, arXiv: 1707.00683. [Online]. Available: http://arxiv.org/abs/1707.00683

[133] E. Perez *et al.*, "FiLM: Visual Reasoning with a General Conditioning Layer," *arXiv:1709.07871 [cs, stat]*, Dec. 2017, arXiv: 1709.07871. [Online]. Available: http://arxiv.org/abs/1709.07871

[134] T. Chen *et al.*, "On Self Modulation for Generative Adversarial Networks," *arXiv:1810.01365 [cs, stat]*, May 2019, arXiv: 1810.01365. [Online]. Available: http://arxiv.org/abs/1810.01365

[135] M. E. Peters *et al.*, "Dissecting contextual word embeddings: Architecture and representation," *CoRR*, vol. abs/1808.08949, 2018. [Online]. Available: http://arxiv.org/abs/1808.08949

[136] N. F. Liu *et al.*, "Linguistic knowledge and transferability of contextual representations," *CoRR*, vol. abs/1903.08855, 2019. [Online]. Available: http://arxiv.org/abs/1903.08855

[137] R. Collobert and J. Weston, "A unified architecture for natural language processing: deep neural networks with multitask learning," in *ICML*, 2008, pp. 160–167. [Online]. Available: https://doi.org/10.1145/1390156.1390177

[138] A. Radford and I. Sutskever, "Improving language understanding by generative pre-training," 2018. [Online]. Available: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[139] T. B. Brown *et al.*, "Language models are few-shot learners," *arXiv*, pp. arXiv–2005, 2020.

[140] A. Merchant *et al.*, "What happens to bert embeddings during fine-tuning?" *arXiv preprint arXiv:2004.14448*, 2020.

[141] P. Rajpurkar *et al.*, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392.

[142] Y. Zhang and Q. Yang, "A survey on multi-task learning," *CoRR*, vol. abs/1707.08114, 2017. [Online]. Available: http://arxiv.org/abs/1707.08114

[143] A. Wang *et al.*, "Can you tell me how to get past sesame street? sentence-level pretraining beyond language modeling," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

[144] S. Wu, H. R. Zhang, and C. Ré, "Understanding and improving information transfer in multi-task learning," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=SylzhkBtDB

[145] T. Yu *et al.*, "Gradient surgery for multi-task learning," *arXiv preprint arXiv:2001.06782*, 2020.

[146] J. Serrà *et al.*, "Overcoming catastrophic forgetting with hard attention to the task," in *ICML*, 2018, pp. 4555–4564. [Online]. Available: http://proceedings.mlr.press/v80/serra18a.html

[147] E. Perez *et al.*, "Film: Visual reasoning with a general conditioning layer," in *AAAI*, 2018.

[148] H. de Vries *et al.*, "Modulating early visual processing by language," in *Advances in Neural Information Processing Systems 30*, I. Guyon *et al.*, Eds. Curran Associates, Inc., 2017, pp. 6594–6604. [Online]. Available: http://papers.nips.cc/paper/7237-modulating-early-visual-processing-by-language.pdf

[149] L. J. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *CoRR*, vol. abs/1607.06450, 2016. [Online]. Available: http://arxiv.org/abs/1607.06450

[150] J. Bingel and A. Søgaard, "Identifying beneficial task relations for multi-task learning in deep neural networks," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 164–169. [Online]. Available: https://www.aclweb.org/anthology/E17-2026

[151] E. Kerinec, C. Braud, and A. Søgaard, "When does deep multi-task learning work for loosely related document classification tasks?" in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 1–8. [Online]. Available: https://www.aclweb.org/anthology/W18-5401

[152] T. Standley *et al.*, "Which tasks should be learned together in multi-task learning?" *CoRR*, vol. abs/1905.07553, 2019. [Online]. Available: http://arxiv.org/abs/1905.07553

[153] Y. Pruksachatkun *et al.*, "Intermediate-task transfer learning with pretrained models for natural language understanding: When and why does it work?" *arXiv preprint arXiv:2005.00628*, 2020.

[154] K. Clark *et al.*, "Bam! born-again multi-task networks for natural language understanding," *CoRR*, vol. abs/1907.04829, 2019. [Online]. Available: http://arxiv.org/abs/1907.04829

[155] B. McCann *et al.*, "The natural language decathlon: Multitask learning as question answering," *arXiv preprint arXiv:1806.08730*, 2018.

[156] R. Aharoni, M. Johnson, and O. Firat, "Massively multilingual neural machine translation," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 3874–3884. [Online]. Available: https://www.aclweb.org/anthology/N19-1388

[157] I. Tenney *et al.*, "What do you learn from context? probing for sentence structure in contextualized word representations," *CoRR*, vol. abs/1905.06316, 2019. [Online]. Available: http://arxiv.org/abs/1905.06316

[158] Y. Tay *et al.*, "Hypergrid: Efficient multi-task transformers with grid-wise decomposable hyper projections," *arXiv preprint arXiv:2007.05891*, 2020.

[159] C. Raffel *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," 2019.

[160] J. Chen *et al.*, "An empirical study of the behavior of active learning for word sense disambiguation," in *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*. New York City, USA: Association for Computational Linguistics, Jun. 2006, pp. 120–127. [Online]. Available: https://www.aclweb.org/anthology/N06-1016

[161] R. Reichart *et al.*, "Multi-task active learning for linguistic annotations," in *Proceedings of ACL-08: HLT*, 2008, pp. 861–869.

[162] F. Ikhwantri *et al.*, "Multi-task active learning for neural semantic role labeling on low resource conversational corpus," in *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, 2018, pp. 43–50.

[163] D. Charles, M. Chickering, and P. Simard, "Counterfactual reasoning and learning systems: The example of computational advertising," *Journal of Machine Learning Research*, vol. 14, pp. 3207–3260, November 2013.

[164] T. Wolf *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *CoRR*, vol. abs/1910.03771, 2019. [Online]. Available: http://arxiv.org/abs/1910.03771

[165] P. He *et al.*, "A hybrid neural network model for commonsense reasoning," in *Proceedings of the First Workshop on Commonsense Inference in Natural Language Processing*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 13–21. [Online]. Available: https://www.aclweb.org/anthology/D19-6002

[166] L. Derczynski *et al.*, "Results of the WNUT2017 shared task on novel and emerging entity recognition," in *Proceedings of the 3rd Workshop on Noisy User-generated Text*. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 140–147. [Online]. Available: https://www.aclweb.org/anthology/W17-4418

[167] D. Q. Nguyen, T. Vu, and A. T. Nguyen, "Bertweet: A pre-trained language model for english tweets," *arXiv preprint arXiv:2005.10200*, 2020.

[168] X. Liu *et al.*, "Adversarial training for large neural language models," 2020.

[169] E. Collins, N. Rozanov, and B. Zhang, "Evolutionary data measures: Understanding the difficulty of text classification tasks," in *Proceedings of the 22nd Conference on Computational Natural Language Learning*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 380–391. [Online]. Available: https://www.aclweb.org/anthology/K18-1037

[170] Y. Bengio *et al.*, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[171] M. Guo *et al.*, "Dynamic task prioritization for multitask learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[172] D. Lepikhin *et al.*, "Gshard: Scaling giant models with conditional computation and automatic sharding," *arXiv preprint arXiv:2006.16668*, 2020.

[173] H. Peng *et al.*, "A mixture of h - 1 heads is better than h heads," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 6566–6577. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.587

[174] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," *CoRR*, vol. abs/1810.04650, 2018. [Online]. Available: http://arxiv.org/abs/1810.04650

[175] Z. Chen *et al.*, "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks," *CoRR*, vol. abs/1711.02257, 2017. [Online]. Available: http://arxiv.org/abs/1711.02257

[176] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," *CoRR*, vol. abs/1705.07115, 2017. [Online]. Available: http://arxiv.org/abs/1705.07115

[177] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[178] M. Joshi *et al.*, "Spanbert: Improving pre-training by representing and predicting spans," *CoRR*, vol. abs/1907.10529, 2019. [Online]. Available: http://arxiv.org/abs/1907.10529