Learning What to Share in Multi-Task Learning

Chantal Amrhein

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2019

Abstract

Multi-task learning is desirable for many applications in natural language processing. It can reduce the size of the machine learning model and can improve performance with increased generalisation and regularisation. Recent work on multi-task learning has focused on creating architectures that can learn where in the network information should be shared between the tasks. This dissertation presents a simpler approach to achieve the same goal: Instead of adapting the multi-task architecture, the network receives access to a task-identifier to learn what information to share between which tasks. The main findings of this work are that task-identifiers are not necessary when multi-task learning is combined with input representations from a current state-of-the-art embedding method, BERT. I show that such embeddings both achieve very high single-task results that are hard to outperform and that they already encode a great deal of task-specific information which renders multi-task learning and task-identifiers less useful. Finally, I argue that the proposed task-identifiers may still have merit in different multi-task learning scenarios and show that they can outperform standard multi-task learning when a less expressive input representation is used.

Acknowledgements

First of all, I would like to thank my supervisors, Gianluca Corrado and Graham Ritchie, for their continuous support throughout my work on this dissertation. It was always a pleasure to meet with you, and I appreciated your valuable feedback and our interesting discussions.

Many thanks to Anastasiia for her friendship, always taking keen interest in my project and her helpful insights and comments.

Additionally, I am extremely grateful to my flatmates, Hsiu-Han and Yanxi. Your company throughout this year was invaluable to me. Thank you for your never-ending support in hard times and sharing my joy in happy moments.

A heartfelt thank you to my best friends back home for their support from afar. You never cease to inspire me.

Last but not least, I want to thank my family who have always believed in me. Thank you for all you taught me and the love you give me.

Without any of you, this work would never have been written. Thank you.

Table of Contents

Chapter 1

Introduction

Over the last few years, deep learning has celebrated massive successes and advancements in many research areas such as computer vision or natural language processing (NLP). What is common to deep learning models is that they have many more parameters that need to be trained compared to previous state-of-the-art approaches. This, in turn, often requires access to vast collections of labelled data to successfully train such models in a supervised way. Additionally, it is essential to have extensive computational resources to train them. This can pose problems when using such models. For example, when only little labelled training material exists or many different models need to be deployed at the same time.

Multi-task learning is a subfield of machine learning that has received more and more attention with the rise of deep learning and has the potential to overcome some of the issues mentioned in the previous paragraph. The term "multi-task model" is used to describe a single system that can perform two or more tasks. This is beneficial considering computational resources as it decreases the number of systems that need to be trained, deployed and maintained. In industrial settings, this is often crucial to save costs and time when working with deep learning models. Similarly, it can have a positive effect on low-resource tasks with little annotated data as more information can be learned from training on additional tasks.

Other benefits of multi-task learning are increased generalisation, regularisation and better representation learning [30]. This is useful in many scenarios. For example, in a study by Talman and Chatzikyriakidis [35], they find that various natural language inference (NLI) architectures only perform well on the specific dataset they were trained on and do not learn how to understand natural language in general. In this case, training a multi-task model on multiple NLI tasks could help generalise across

different datasets and extract better representations that facilitate natural language understanding on new data.

These properties make multi-task learning an interesting research area that is worth exploring in more depth. Previous work mostly applied multi-task learning to a new set of tasks or investigated on what types of tasks it performs well [7, 33, 17, 18, 27, 1, 21, 25, 2]. The general outcomes of these studies are that multi-task learning does not always result in improved performance and that its success may depend on many different factors. For most of these experiments, the multi-task learning architecture is designed to share some of the lower layers and have one or more task-specific layers on top. This means that in the shared layers the network cannot distinguish between different tasks and assumes the input is task-independent. This can work well as a form of regularisation but the parts of the network that are to be shared are determined before training. Deciding on the position of shared layers in advance may stem from a misconceived assumption that this is the best way to learn in a multi-task scenario.

More recent research has focused on developing multi-task networks that can learn what parts of the network should be shared between tasks [31, 16]. The proposed architectural solutions have more trainable parameters and are more complicated than standard multi-task learning architectures. Based on the same idea that the multi-task network should be able to learn what information can be shared between which tasks, I propose a simpler strategy to achieve this. Instead of adapting the architecture with additional parameters that control sharing, I simply give the network access to a task-identifier as part of the input. Not only will such an approach be much simpler and have less parameters than previous work, it also does not assume anything about how the model should be using the information about the tasks. The model has full control over the task-identifier and can potentially choose to completely ignore it.

The proposed approach is motivated by a recent success in multilingual neural machine translation: a single encoder-decoder translation model was trained to translate between multiple source and target languages by using a target language identifier as part of the input sequence [13]. In this dissertation, I apply the same idea to multi-task learning and compare the proposed task-identifier model to a standard multi-task learning approach with fixed shared and task-specific layers. The project investigates the following two research questions:

- **Does multi-task learning improve the performance compared to single-task baselines?**
  The obtained results suggest that the answer to this question is not straightforward. Even though in some experiments the multi-task model can improve the overall performance over the single-task baselines, this is not the case for a state-of-the-art embeddings method, BERT. I show that such an input representation results in very high single-task performance and is extremely hard to outperform with multi-task learning. Furthermore, I present an analysis that links these unsuccessful multi-task results to task-specific information that is encoded with BERT.

- **Does multi-task learning with a task-identifier as part of the input outperform the standard multi-task learning architecture?**
  I find that a current state-of-the-art embedding method, BERT, already encodes a great deal of task- / dataset-specific information in the input representation for the multi-task learning network. This makes additional task-specific information less useful and explains why the task-identifier multi-task learning experiment performs comparatively to the standard multi-task learning approach. However, I show that a task-identifier has merit when the input representation to the network is less expressive. Nevertheless, the conflicting results make it hard to find a general answer to this question and more experiments are needed. Thus, I present ideas for further research on task-identifiers for multi-task learning.

This dissertation is structured as follows: Chapter 2 presents the relevant background to the project and discusses previous efforts in multi-task learning. It also introduces the state-of-the-art embeddings method, BERT, that is investigated in this work. In Chapter 3, I present the datasets used for experiments. Next, Chapter 4 explains my methodology and discusses the implemented multi-task architecture and the training process in greater detail. In Chapter 5, I present the results of my experiments and discuss them with respect to the research questions and other work. Finally, Chapter 6 concludes this dissertation and highlights ideas for future work.

Chapter 2

Background

2.1 Multi-Task Learning in NLP

Multi-task learning is a subfield of machine learning which is motivated by the belief that humans transfer previous knowledge to any new task they have to solve and that they never learn in isolation. This same concept can be found in two machine learning strategies known as transfer learning and multi-task learning. Whenever knowledge, such as model parameters, is transferred from one task model to another, this is known as transfer learning. Whenever two or more objective functions are optimised within the same model, this is considered multi-task learning. Even before neural networks became increasingly popular due to the growing amount of available training data and computing resources, Caruana [4] argued that neural networks are particularly suited for multi-task learning problems. Since they learn hidden representations of the data, these representations can be shared between multiple tasks. This has a number of positive effects on the performance of multi-task models such as increased generalisation, regularisation and better representation learning [30].

One of the first works to apply multi-task learning to NLP in the context of deep learning was by Collobert and Weston [7]. They argued that, in traditional NLP, outputs from certain tasks such as part-of-speech tagging and chunking are often given as input features to downstream tasks such as semantic role labelling. This comes with the drawback that any errors generated in earlier stages of this "pipeline" are propagated to later tasks. Their idea was to train a multi-task model which can learn how to solve a number of NLP tasks at the same time using deep learning. Their application of multi-task learning improved generalisation and achieved better performance for all involved tasks; for semantic role labelling it even established a new state-of-the-art result. This

success sparked further interest in the field.

As deep learning became more prominent in the field of NLP over the years, so did multi-task learning: Liu et al. [17] proposed three multi-task architectures that can be used for training recurrent neural networks (RNN). RNNs are well-suited to model sequences with neural networks as they have connections to hidden states from previous time steps [12]. This allows them to take into account the history of a sequence when making a new prediction, which makes them useful for many NLP tasks [41]. The proposed multi-task approaches outperformed several baselines on different text classification tasks. Søgaard and Goldberg [33] showed that task-independent output layers do not necessarily need to be placed on the same level within the network. Some tasks can be ordered in a hierarchical way such as part-of-speech tagging before chunking and chunking before semantic role labelling following an increasing order of difficulty. In this scenario, the authors show that it is useful to have the task-specific output layers for lower-order tasks at lower layers in the network whereas the task-specific output layers for higher-order tasks are placed deeper in the architecture. In this way, the network can mirror the hierarchical order of the tasks. This approach has become known as low supervision multi-task learning. Other notable works have applied multi-task learning to NLP tasks including semantic classification and information retrieval [18] as well as named entity recognition [27].

In general, most of these applications of multi-task learning are cases of hard parameter sharing [30], i.e. the parameters of the shared layers are shared between all involved tasks. This is in contrast to soft parameter sharing [30] which means that parameters are initialised and updated for every task individually but a regularisation term is used to force parameters to be similar for the involved tasks. Hard-parameter sharing is simpler to implement and involves less parameters which is why it is often chosen for multi-task learning applications. However, it does not always prove to be successful which is discussed in the next section.

2.2 Challenges of Multi-Task Learning

Although multi-task learning has been proposed for a wide range of NLP tasks, it has had mixed success. While it is interesting to apply multi-task learning to new tasks and datasets, it is equally important to know under what conditions multi-task learning performs well. There are a number of studies that investigated this question. Benton et al. [1] employ a multi-task learning model for predicting suicide risk and mental health

conditions which shows significant gains over single-task baselines. Two findings are particularly interesting: First, multi-task learning leads to the largest gains in performance for tasks where only little training data is available. Second, it is important to choose auxiliary tasks carefully in a multi-task setting as this can largely influence the performance of the model.

Martínez and Plank [21] test how information-theoretic measures such as the entropy of a task's label distribution influence the performance of multi-task learning. Their findings show that for several sequence labelling tasks multi-task learning with auxiliary tasks is not always improving over single-task baselines. One visible trend is that auxiliary tasks with a small number of different classes and uniform label distributions are more useful in a multi-task learning setting. Similarly, Mou et al. [25] investigate how transferable neural networks are in various sentence and sentence pair classification scenarios. They distinguish between transfer learning through parameter initialisation and multi-task learning and find that these two methods perform comparatively. One of their main conclusions is that the semantic relatedness of the tasks involved is important for the success of multi-task learning: the performance on datasets from semantically similar or equivalent tasks is improved with multi-task learning while this is not the case for semantically dissimilar tasks.

Bingel and Søgaard [2] present an interesting experiment to find key factors needed for successful multi-task learning. First, they train single-task models and multi-task models for ten different NLP tasks. Next, they train a logistic regression model to predict the gain of multi-task learning over the single-task model. The logistic regression model is trained using features extracted from dataset characteristics such as dataset size or number of labels and from the single-task training behaviour such as gradients of the learning curve. The results suggest that multi-task learning gains can indeed be predicted from these features. The most useful features are the single-task learning curves which suggests that multi-task learning can help tasks that suffer from the local minima problem. Another helpful feature is the label entropy. Interestingly, the difference in size between tasks is not a reliable predictor which suggests that the finding of Benton et al. [1] is not universally true.

In this dissertation, the focus is not on finding tasks on which multi-task learning performs well but rather on enhancing multi-task learning in a way that allows learning what information to share between tasks. Ideally, this would make multi-task learning more beneficial in scenarios where it has previously only shown limited success. The proposed strategy to achieve this involves passing a task-identifier to the network

as part of the input representation. Previous research that focused on learning with additional input information is presented in the next section.

2.3 Learning with Additional Identifiers

The idea of incorporating additional knowledge in deep learning models is very prominent in neural machine translation (NMT). Sennrich et al. [32] use an additional input token to mark if a translation from English to German should be informal or formal. They successfully show that this token can control the level of politeness in the output of the machine translation system. Similarly, Johnson et al. [13] use an additional input token to train a multilingual NMT system. The token marks the desired target language of the output translation. This allows the use of a single translation model for translating between many language pairs and even "zero-shot" pairs which have not been seen during training. Following this success, Chu et al. [5] use an identifier for different domains to perform domain adaptation in NMT in a simple way.

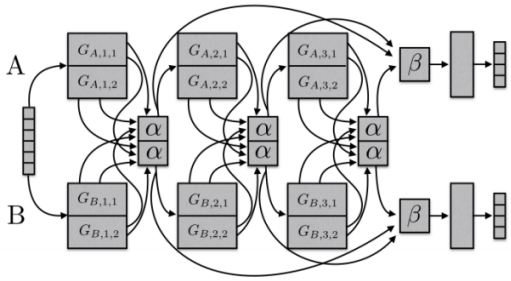

Figure 2.1: "A SLUICE NETWORK with one main task A and one auxiliary task B. It consists of a shared input layer (shown left), two task-specific output layers (right), and three hidden layers per task, each partitioned into two subspaces. α parameters control which subspaces are shared between main and auxiliary task, while β parameters control which layer outputs are used for prediction." (Figure and description taken from Ruder et al. [31]: Figure 1)

In multi-task learning, the inclusion of additional task information as part of the input has, to the best of my knowledge, not been explored so far. A related approach are sluice networks [31]. Based on the finding that multi-task learning is not always beneficial, Ruder et al. developed a model that learns additional parameters which control the information that is shared between tasks. By adapting these parameters during training, the network can learn in what layers and what parts of these layers the

information should be shared or kept task-specific. Figure 2.1 shows a sluice network in more detail. The authors evaluate their framework on a broad set of sequence-tagging tasks and compare it with previous multi-task learning strategies such as hard parameter-sharing, low supervision [33] and cross-stitching [24]. The sluice network significantly outperforms all other strategies in almost all tasks. As discussed in earlier sections hard parameter-sharing and low supervision multi-task learning (which simulates a hierarchy of tasks) involve setting which parameters are shared before training the network. In this respect, cross-stitching is more similar to sluice networks as they also learn which parameters are shared. In fact, sluice networks are a generalisation of cross-stitch networks which were introduced specifically for multi-task learning with convolutional neural networks [24]. While sluice networks outperform other approaches, they are a lot more complex to understand and involve many more parameters than simpler multi-task learning architectures. It is therefore interesting to see if simply adding task-related information as part of the input representation can improve multi-task learning as well.

Another architectural solution for selective sharing is proposed by Liu et al. [16]. Their multi-task learning solution for text classification consists of task-specific layers with long short-term memory (LSTM) [10] units; a special kind of RNNs which consists of a memory cell and three trainable gates that control the information flow. Instead of sharing whole layers of LSTM units, Liu et al. designed task-specific LSTM units that have access to a shared memory cell. Every LSTM unit can selectively read from and write to this shared memory cell by controlling a trainable gate. In contrast to standard multi-task learning architectures, this memory-enhanced LSTM can learn how much information is shared between tasks. Unlike the approach proposed in this thesis, there is no distinction between information coming from different tasks in the shared memory cell. Essentially, the LSTM can "decide" to access the shared memory but it cannot selectively use the information coming from only one other task.

Most similar to using a task-identifiers in a more constricted multi-task learning scenario is work by McCann et al. [22]. They train a single system without any task-specific components that can perform 10 different NLP tasks. To achieve this, all involved tasks are reformulated as sequence-to-sequence question answering tasks where the desired task is given as part of the question. For example, for a sentiment analysis task the question would be "Is this sentence positive or negative?" and the corresponding sentence would be given as additional context to the network. The expected answer would then be either "True" or "'False". While such a model is desirable in

theory, its current performance does not reach single-task baselines. In this dissertation, I explore a more standard multi-task learning setup with task-specific components and analyse the effect of an additional task-identifier.

2.4 Contextual Embeddings

Word embeddings [23] have become a crucial instrument for NLP over the last years. They replaced traditional bag-of-word features as the standard input to many NLP models. Word embeddings encode words into vectors that are closer together in the vector space if the words commonly appear in similar context. However, they do not take into account the context of where the word appears in a specific sentence. Homographs such as "does" (in the sense of multiple female deer) and "does" (in the sense of the third person singular form of the verb "do") are assigned the same vector representation with standard word embeddings. This drawback has sparked interest in developing models that can learn "contextual" word embeddings.

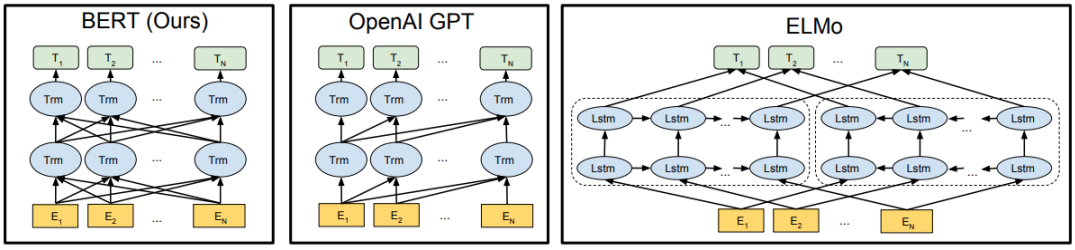

Figure 2.2: Comparison of BERT, OpenAI GPT and ELMo. (Figure taken from Devlin et al. [8]: Appendix, Figure 3)

One of the first contextual word embedding models is called ELMo [26]. Figure 2.2 shows the architecture of ELMo on the right. To generate contextual word embeddings, ELMo concatenates the output of an independent block of stacked left-to-right LSTM [10] layers and a block of right-to-left LSTM layers. Shortly after ELMo was published, two other contextual word embedding models were developed: OpenAI GPT [28] (in the middle) uses stacked left-to-right Transformer layers to encode the input instead of LSTM layers. The Transformer [36] is a novel architecture that is intended to perform better on long-range dependencies where RNNs often struggle. Transformers allow parallel computation during training since there are no longer any recurrent structures. Instead, a Transformer layer computes a representation of its

input only using self-attention [36] and positional encoding. In contrast to OpenAI GPT, BERT [8] (on the left) is designed to be deeply bidirectional because it is built with stacked bidirectional Transformer layers. BERT can be seen as a combination of ELMo with its shallow bidirectionality and OpenAI GPT with its transformer layers. Despite their differences, all of these contextual word embeddings are a function of the input sequence. One drawback when using contextual word embeddings is that they involve more than a simple matrix lookup; all input sequences have to be run through an encoder to obtain the embeddings.

Contextual word embeddings and their pretraining architectures have since been used in many NLP applications. Liu et al. [19] argue that unsupervised language model pretraining used for these embeddings and multi-task learning are complimentary in most NLP scenarios. They propose a multi-task learning architecture on top of BERT and fine-tune it for a number of given tasks. The parameters of BERT are shared between the tasks and each task has an individual output layer on top. This multi-task setup outperforms fine-tuned single-task models on the same tasks. The success of this work has inspired the use of BERT in this thesis. Instead of fine-tuning BERT on multiple tasks, the input sentences will be run through BERT and taken as input features for the multi-task model. Feature-based BERT representations do not perform as well as fine-tuning BERT [8] but due to limited computational resources and a short time frame, fine-tuning BERT for all experiments is not feasible in this dissertation.

Two other important, concurrent works are by Houlsby et al. [11] and Stickland and Murray [34]. Both propose a similar alternative to standard transfer learning techniques such as fine-tuning or using the output of a contextual word embeddings model as input features to another model. After an initial model is trained, the adaptation is done with inserted adaptor modules: Between the layers of the original network, trainable layers are inserted and trained on the new task while the original parameters remain frozen. While the implementation of these adaptor modules differs slightly between the two proposed methods, they achieve the same goal. The number of parameters that need to be trained in order to use a large pretrained model for a new task is significantly reduced. Both sets of authors tested this approach with a pretrained BERT model and found that training only the adaptors on a new task while all BERT parameters are frozen can bring almost the same performance as fine-tuning all parameters of the BERT model. While Houlsby et al. [11] trained their adaptors specifically for one task, Stickland and Murray [34] reduced the trainable parameters even further by assuming a multi-task learning scenario and sharing the adaptors between tasks.

Chapter 3

Data

3.1 GLUE Benchmark Datasets

The GLUE benchmark is a collection of "nine sentence or sentence-pair natural language understanding tasks, built on established annotated datasets and selected to cover a diverse range of text genres, dataset sizes, and degrees of difficulty" [37]. For the experiments in this thesis, a subset of four GLUE tasks is selected that represent a diverse collection of dataset properties as described later in this chapter: MNLI (Multi-Genre Natural Language Inference), RTE (Recognizing Textual Entailment), QQP (Quora Question Paraphrases) and MRPC (Microsoft Research Paraphrase Corpus). Since the original GLUE test sets are provided without labels for fair evaluation, a new random train, development and test split was created from the original train and development set. Table 3.1 gives an overview of these splits. For every dataset, roughly 80% of the available labelled data is used for training, 10% for validation and another 10% for testing. Two of the chosen tasks are related to natural language inference: MNLI and RTE are both textual entailment tasks. The other two tasks, QQP and MRPC, are paraphrase identification tasks.

In terms of size, QQP and MNLI are comparatively high-resource tasks with roughly 300,000 sentence pairs for training. In contrast, RTE and MRPC are low-resource tasks with about 100 times less training data. While all tasks are classification tasks it is noteworthy that MNLI is a 3-class problem whereas all other tasks are binary classification problems. The label distribution in MNLI and RTE is balanced. QQP and MRPC are unbalanced datasets. There are 63% negative examples and only 37% positive examples in QQP while in MRPC roughly two thirds are positive and only one third negative examples. The datasets are described in more detail in the rest of this chapter.

Dataset	Task	Train	Dev	Test
MNLI	ENT	312,198	40,250	40,250
		80%	10%	10%
RTE	ENT	2,213	277	277
		80%	10%	10%
QQP	PI	323,430	40,430	40,430
		80%	10%	10%
MRPC	PI	3,260	408	408
		80%	10%	10%

Table 3.1: Summary of the datasets. Sizes are given as number of sentence pairs. ENT = entailment task, PI = paraphrase identification task

QQP is a dataset built with question pairs from the question-and-answer website "Quora[1]". The questions are rather short with an average sentence length of 12.76 tokens. For each question pair, it needs to be determined whether they are paraphrases of each other, i.e. whether they are semantically equivalent. Unfortunately, the labels are known to be noisy with a small fraction of clear paraphrases being labelled as no paraphrases and vice versa.[2] Given its origin, the language of this dataset is very conversational. Here are two examples that illustrate the content of this dataset:

(1) QQP - paraphrase example

`Do dogs have any sense of what part of the week it is?`

`Does my dog know what day of the week it is?`

QQP - no paraphrase example

`How do I unlock my Yahoo Mail account?`

`How do you disable a Yahoo account?`

MRPC [9] is another paraphrase identification dataset. It is constructed with sentences from online news articles.[3] As expected for this domain, the average sentence length is much longer than for QQP with 22.34 tokens per sentence. Again, here are two sentence pairs along with their labels taken from the dataset:

(2) MRPC - paraphrase example

`About 1,557 genes on chromosome 6 are thought to be functional.`

[1] `https://www.quora.com`

[2] `https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs`

[3] `https://www.microsoft.com/en-us/download/details.aspx?id=52398`

`The remaining 1,557 genes are believed to be all functional.`

MRPC - no paraphrase example

`The blaze was only 5 percent contained early Monday.`

`The blaze, started by lightning June 6, is considered 95 percent contained.`

MNLI [39] is built from ten different domains of spoken and written text. Through crowd-sourcing, every sentence pair is labelled with information about the textual entailment of the two sentences. The task is to identify whether the second sentence entails the first one, contradicts it or whether there is a neutral relation. The dataset has been shown to be more difficult for machine learning models than a similar textual entailment dataset, SNLI [3]. The average sentence length is 17.01 tokens. Below are example sentences for every label class:

(3) MNLI - entailment example

`The sacred is not mysterious to her.`

`The woman is familiar with the sacred.`

MNLI - contradiction example

`But I thought you'd sworn off coffee.`

`I thought that you vowed to drink more coffee.`

MNLI - neutral example

`History defies laws.`

`Laws are weak when talking about the past and the future.`

The RTE dataset is another textual entailment identification task. The dataset was built using multiple smaller datasets from competitions on textual entailment (for details see [37]). The source texts are a mix of newspaper articles and texts from Wikipedia[4]. The average sentence length is 30.43 tokens which is almost twice as long as MNLI and the highest average sentence length of the four chosen datasets. In contrast to MNLI, RTE is a binary classification problem that only distinguishes between entailment and not entailment. Here is an example sentence pair for every class label:

(4) RTE - entailment example

`By clicking here, you can return to the login page.`

`Click here to go back to the login page.`

RTE - not entailment example

`The Statue of Liberty is so big it had to be built in 300 sections.`

`The Statue of Liberty was built in the year 300.`

[4] `https://www.wikipedia.org/`

Chapter 4

Methods

4.1 Input Representation

4.1.1 BERT Sentence Encoding

For the experiments described in this thesis, I extract sentence representations using a pretrained BERT model[1]. The original output of BERT contains a representation for a special "CLS" token at the beginning of each sentence. In previous work, this token was trained to represent the whole sentence for classification tasks by fine-tuning the pretrained BERT model on a selected dataset [8, 19]. In this dissertation, BERT is not fine-tuned but rather its output is used as input features for a simple classification network. Therefore, the "CLS" token cannot be used as a representation of the whole sentence since the training objective during pretraining is different for this token than the tasks investigated in this work. To achieve a representation of the whole sentence, I mean-pool over all BERT-encoded token vectors. For this, I use "bert-as-service" [40] which offers a fast sentence encoding process using a pretrained BERT model. In early experiments, I compared the performance of mean-pooled and max-pooled BERT vectors as well as using the "CLS" token as a representation for the whole sentence. I found that mean-pooled sentence representations worked best which is in line with an analysis by Xiao who developed "bert-as-service".[2] Here is an example of how a sentence pair is encoded into a vector with a pretrained BERT model:

[1] https://github.com/google-research/bert#pre-trained-models

[2] https://github.com/hanxiao/bert-as-service#q-so-which-layer-and-which-pooling-strategy-is-the-best

(5) Original Sentence Pair:

`How do I unlock my Yahoo Mail account?`

`How do you disable a Yahoo account?`

BERT Input Format:

`How do I unlock my Yahoo Mail account? ||| How do you disable a Yahoo account?`

Internal BERT Tokenization:

`['[CLS]', 'how', 'do', 'i', 'unlock', 'my', 'ya', '##ho', '##o', 'mail', 'account', '?', '[SEP]', 'how', 'do', 'you', 'di', '##sable', 'a', 'ya', '##ho', '##o', 'account', '?', '[SEP]']`

Original BERT Output Token Representation Shape:

`(1, 25, 768)`

Mean-pooled BERT Output Sentence Representation Shape:

`(1, 768)`

To use BERT as an encoder, it is important to use the same SentencePiece [15] model that was used during the pretraining to split the input into smaller subwords. This effectively makes BERT an open vocabulary approach, meaning that all words can be encoded and there are no unknown words. Proper names such as "Yahoo" in the example sentence pair are often split up into smaller units. BERT[3] encodes every one of these subword units into a 768-dimensional vector. Mean-pooling over all the subword tokens results in one 768-dimensional vector that represents the whole sentence pair. This is the input representation of the sentence pairs used in all of my experiments.

4.1.2 Task-Identifier Embedding

This dissertation investigates the benefit of giving a task-identifier as additional information to train multi-task learning models. This is very simple in a neural network model. Either it can be included as an additional input token or it can be represented as part of the vector representation of the whole input. For my experiments, I decided on the second option since the pretrained BERT model was not trained to encode a task-identifier token.

To add the task-identifier to the vector representation of the input, a one-hot vector task-identifier representation is first passed through an embedding layer. The one-

[3] I used BERT-Base Cased which has around 110 million parameters.

hot vector is a vector that has as many dimensions as the total number of involved tasks. For a specific task, all vector elements will be set to 0 except for the one that corresponds to the ID of the task. This dimension will be set to one. When multiplied with the embedding layer weight matrix only the weights for that task will be applied since all other elements are set to 0. The weights of the embedding layer are trainable and will be updated with the other components of the network.

The 16-dimensional task-identifier embedding is then concatenated with the 768-dimensional sentence representation coming from BERT which was described in the previous section. For the experiments without the task information, the task-identifier is simply set identical for all the tasks. This concatenated vector of the task-identifier and the sentence pair is the input to the architecture described in the next section.

4.2 Multi-Task Learning Architecture

Multi-task learning can either be implemented with hard parameter sharing or soft parameter sharing [30]. In hard parameter sharing, the parameters of lower-level layers are shared between tasks while higher-level layers have task-specific parameters. In soft parameter sharing, no parameters are explicitly shared but there is a regularisation term that ensures that the parameters of the different task-specific models are similar to each other. In this thesis, I develop a hard parameter sharing architecture. Figure 4.1 shows a schema of the multi-task learning architecture.

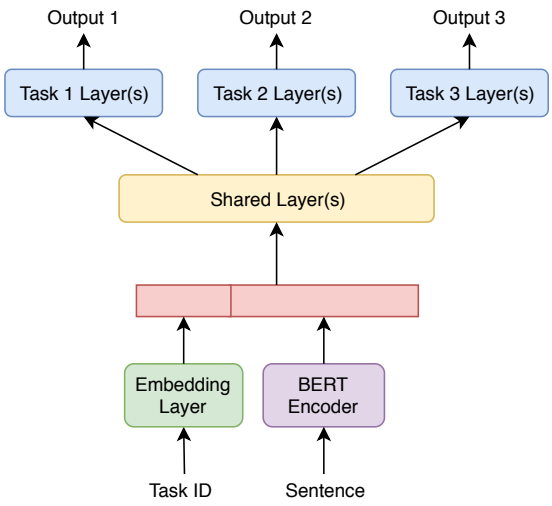

Figure 4.1: Visualisation of the multi-task learning architecture used in this dissertation.

As described in the previous section on the input representation, every sentence pair is encoded with BERT (purple) and the task-identifier is passed through an embedding layer (green). Then, both are concatenated to form the input representation to the architecture (red). This input vector is then passed through a number of fully-connected, non-linear hidden layers with ReLU activation function that build the shared layer(s) (yellow). On top of the shared layer(s), for every involved task, there is one or more fully-connected task-specific layers (blue) with ReLU activation function. The parameters of these layers are all task-specific. In the single-task models, the parameters in the otherwise shared layer(s) are also task-specific. For every task, the last of the task-specific output layers is a linear feed-forward projection whose output is passed to the softmax function to obtain probabilities for every possible class. The softmax function defines the probability for a class c given the output of the network o:

$$Pr(c|o) = \frac{\exp(o_c)}{\sum_k^C \exp(o_k)} \tag{4.1}$$

4.3 Multi-Task Learning Training

For my experiments, I train my models with task-specific mini-batch training. This means that task-specific mini-batches for every involved task are created and then randomly shuffled between all tasks. This follows the training procedure by Liu et al. described in [19]. Algorithm 1 (on the next page) describes the task-specific mini-batch training in more detail.

Since all involved tasks are classification tasks, I use the cross-entropy loss function. The formula is shown in Algorithm 1 on line 11, where C is the total number of output classes for task t, N is the batch size, i is the current example in the batch, $\mathbb{1}(B_{t_i}, c)$ is 1 if class label c is the correct classification for B_{t_i} or 0 otherwise and $Pr(c|o_{t_i})$ is the softmax probability for class c given network output o_{t_i} as defined in Equation 4.1 in the previous section. The losses are renormalised according to the number of classes to avoid having much higher losses for MNLI which is the only non-binary problem. All losses are directly applied without accumulating or combining the losses of different tasks in some way, e.g. by weighting them proportionally to the training dataset size.

Algorithm 1 Task-Specific Mini-Batch Training (inspired by [19])

1: Initialise model parameters Θ randomly.
2: Initialise time step counter $j = 0$ and Adam exponential moving averages $m = 0,\ v = 0$.
3: Load all datasets 1, 2, ... T.
4: **while** not early stopping **do**
5: **for** t in 1, 2, ..., T **do**
6: Pack dataset t into mini-batches: D_t
7: **end for**
8: Merge all mini-batches of all datasets: $D = D_1 \cup D_2 ... \cup D_T$
9: Shuffle D
10: **for** B_t in D **do**
11: Forward-propagate B_t through the network: o_t # B_t is a mini-batch of task t
12: Compute cross-entropy loss: $L(\Theta) = \frac{1}{C}\ \frac{1}{N} - \sum_i^N \sum_c^C\ \mathbb{1}(B_{t_i}, c)\ \log(Pr(c|o_{t_i}))$
13: Compute gradient: $\nabla(\Theta) = \frac{\partial\, L(\Theta)}{\partial\, \Theta}$
14: Update model parameters with the Adam update rule [14]:
$j = j + 1$
$m = \beta_1\ m + (1 - \beta_1)\ \nabla(\Theta)$
$\hat{m} = \frac{m}{(1-\beta_1^j)}$
$v = \beta_2\ v + (1 - \beta_2)\ \nabla(\Theta)^2$
$\hat{v} = \frac{v}{(1-\beta_2^j)}$
$\Theta = \Theta - \frac{\eta}{\sqrt{\hat{v}}+\varepsilon}\hat{m}$
15: **end for**
16: **end while**

For optimisation, I use Adam [14] as defined in line 13, where η is the learning rate, β_1 is set to 0.9, β_2 is set to 0.999 and ε is set to 10^{-8}. The training is stopped by early stopping when the loss on the development set has not decreased over 3 epochs. Early stopping on the loss has proven more stable than on other metrics in early experiments. In the multi-task case, the mean loss over all tasks is used as the stopping criterion.

Chapter 5

Results

5.1 Baselines

As a baseline for my multi-task learning experiments, I train a single-task model for every chosen task. I use the same architecture as described in Chapter 4 to train the single-task models. In contrast to the multi-task learning scenario, all parameters are task-specific. To ensure I compare the later experiments against strong baselines, I tune the hyper-parameters for all tasks. I test different values for the learning rate, batch size, dropout, the hidden size of the first layer and early stopping patience by performing a grid search and evaluating the results on the development set. Based on these experiments, I find that the following hyper-parameters shown in Table 5.1 work best for the specific datasets. An early stopping patience of 3 worked well across all tasks. On top of the first hidden layer, there is another hidden non-linear layer with 64 hidden units followed by a linear projection layer which is used as the input to the softmax operation. I tried using wider and/or deeper networks with more hidden layers but saw no significant improvements.

	Learning Rate	**Batch Size**	**Dropout**	**Hidden Size #1**
MNLI	5×10^{-5}	64	0.2	256
RTE	3×10^{-5}	16	0.1	128
QQP	5×10^{-5}	64	0.2	256
MRPC	3×10^{-5}	16	0.1	128

Table 5.1: Best-performing hyper-parameters for every task.

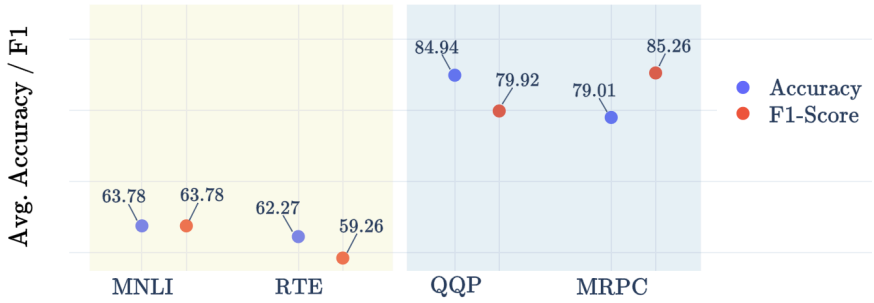

Figure 5.1: Average accuracy and F1-score of the single-task baselines on the test set. Results are computed over three independent runs.

The results for the described single-task models on the test sets can be seen in Figure 5.1. Learning curves and tabular results for the models can be found in the Appendix. For MNLI, QQP and MRPC accuracy and F1-score behave as expected since MNLI is a balanced dataset, QQP contains more negative labels and MRPC more positive labels. Interestingly, the F1-score for RTE is much lower than the accuracy despite it being a balanced dataset. A quick look into the predictions on the test set shows that the model generates more true negatives, false positives and false negatives than true positives. This suggests that identifying true entailment in this dataset is harder than in the MNLI dataset. Another observation is that the textual entailment tasks (marked in yellow) seem to be much harder given the BERT sentence embedding than the paraphrase identification tasks (marked in blue). This result does not follow most entries in the GLUE leaderboard[1] where the performance differences between these two task types are often much less prominent. Generally, the presented single-task baselines do not reach state-of-the-art results which reflects the restrictions of whole sentence representations compared to fine-tuning BERT for example [8].

5.2 Multi-Task Learning Approaches

5.2.1 Standard Multi-Task Learning

The first multi-task learning experiment is testing the standard multi-task learning architecture on the chosen datasets. Again, I perform hyper-parameter tuning with a grid search on the development set. I find that task-specific learning rates and batch sizes

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

help the model to learn more smoothly. The tuned learning rates and batch sizes for the individual tasks (Figure 5.1) also work best in the multi-task setting. Furthermore, the model performs best with a dropout of 0.2 and 256 hidden units in the first layer. Like the second hidden layer for the single-task models, the task-specific layers on top of the shared (first) hidden layer have 64 hidden units followed by a linear projection layer and softmax. Again, I tested the effect of using wider and/or deeper networks with more hidden layers but saw no significant improvements. As described in Chapter 4, the task identifier is set identically for every task in the standard multi-task learning experiments.

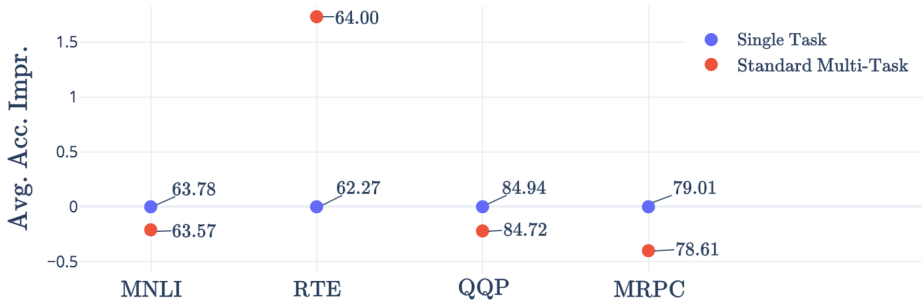

Figure 5.2: Average absolute improvement in accuracy of the standard multi-task experiments over the single-task experiments. Results are computed over three independent runs.

The improvement in accuracy of the standard multi-task learning experiment over the baseline can be seen in Figure 5.2. The corresponding F1-scores are not presented here as they show the exact same behaviour. Learning curves and tabular results for the models can again be found in the Appendix. The standard multi-task learning models (red) perform worse than the single-task models (blue) for most datasets even though the gap between them is very small. For the two larger tasks, MNLI and QQP, this difference is consistent with a standard deviation of 0.065 and 0.101 respectively. The standard deviations of the improvement in accuracy for the two low-resource tasks, RTE and MRPC, are much higher with 0.568 and 0.367 respectively. This shows that the performance of the standard multi-task model is not as stable on these tasks which makes it hard to draw any general conclusions. However, for RTE, the multi-task model performs much better than the single-task model even taking into account the high standard deviation. Other multi-task experiments on fine-tuning BERT have also reported large improvements on this task compared to a single-task baseline [19, 34, 6]. While it is possible, that this improvement in my experiments is caused by sharing

some parameters between tasks, there may also be another reason for this. RTE is the smallest task out of the four chosen datasets. The development and test set only contain 277 sentences each. This can lead to validation overfitting when early stopping on the development set. Meaning that training is stopped at a point that is beneficial for the development set but not for the test set. In this case, multi-task learning can be helpful, as the early stopping criterion is not solely based on the loss for that task but rather all tasks.

Initially, I suspected that my simple multi-task setup is the reason that multi-task learning did not improve over the single-task baselines. Therefore, I ran many experiments testing more elaborate multi-task learning strategies. Here is a list of some approaches I tested:

- other optimisers than Adam
- increasing the width and depth of the network
- learning rate schedulers and weight decay
- weighting the losses of the different tasks
- batch normalisation
- oversampling the smaller tasks instead of task-specific mini-batch training
- individualising hyper-parameters for the task-specific layers such as dropout
- token-based BERT representations with a bidirectional LSTM sentence encoder instead of mean-pooled, sentence-based BERT representations

Additionally, I tried other combinations of different datasets from the GLUE benchmark as well as other well-known datasets. I also created artificial low-resource settings and conducted experiments only using closely related tasks for multi-task learning. In none of these experiments, did I see a significant gain in accuracy or F1-score when using standard multi-task learning compared to the single-task baselines.

There is no other research that used BERT-encoded sentence representations as a feature-based input for multi-task learning. However, I can compare my results to other work that has investigated multi-task learning by fine-tuning BERT on GLUE datasets. Liu et al. [19] initially proposed this and reported mostly consistent improvements over their single-task fine-tuned BERT baseline. More recent research focused mainly on different experiments such as using adaptors to reduce the number of parameters when fine-tuning BERT [34] or using knowledge distillation to teach multi-task models with single-task model teachers [6]. Both of these works used a standard multi-task learning setup for fine-tuning BERT as one of their baselines. In their experiments, the multi-task model also showed a worse performance compared to the single-task baseline as

in my experiments. From these findings, I conclude that it is generally difficult to outperform a single-task baseline with a multi-task learning approach when BERT is involved in some form. This may change when task-specific information is introduced in the shared layers. Experiments with task-identifiers are discussed in the next section.

5.2.2 Multi-Task Learning with Task-Identifier

Since the standard multi-task approach did not show any consistent improvements over the single-task baselines, it is worthwhile to explore how the multi-task model with a task-identifier as part of the input performs. For this experiment, the hyper-parameters from the earlier multi-task experiment are kept identical for comparability. The only difference between these two models is that, for every task, the model will now receive an individual task-identifier whereas before the same one was used for all tasks. This allows the shared layer to learn some additional information about the tasks it solving.

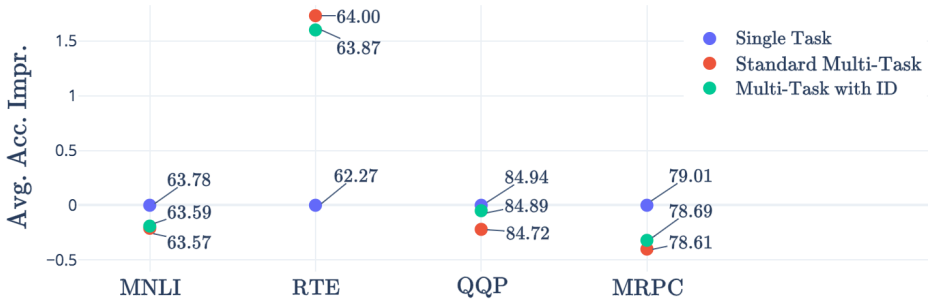

Figure 5.3: Average absolute improvement in accuracy of the multi-task experiments over the single-task experiments. Results are computed over three independent runs.

Figure 5.3 presents the results of the multi-task learning with task-id compared to the other experiments. Again, the results for the F1-score show the same behaviour as the accuracy and are not included here. Learning curves and tabular results for the models can be found in the Appendix. Multi-task learning with the task-identifier (green) performs very similarly to standard multi-task learning (red) in terms of the average improvement in accuracy over the single-task baselines (blue). Considering the standard deviation of the improvement, the two larger tasks have again smaller standard deviations with 0.095 for MNLI and 0.054 for QQP. For the two low-resource tasks, RTE and MRPC, the standard deviation is again higher with 0.662 and 0.971 respectively. This marks an increase over the standard deviations shown in standard multi-task training. This suggests that multi-task learning with the task-identifier gives

less stable performance for low-resource tasks than the standard multi-task learning approach. Again for RTE, the task-identifier model shows a similar gain over the baseline compared to the standard multi-task model including the standard deviation of the improvement. As mentioned before, it is unclear whether this improvement comes from sharing some parameters between the different tasks or less validation overfitting on the small development set.

The results of the experiments with the task-identifier suggest that having this additional input information is not helpful for multi-task learning. This contradicts my hypothesis that such additional information could improve multi-task learning. However, it is important to remember that the input for these experiments has already been propagated through a very powerful model. It remains unclear what influence the BERT encoding has on the results of my multi-tasking experiments. The next section investigates this question in more detail.

5.3 BERT as Input for Multi-Task Learning

Multi-task learning did not show consistent improvements over the selected GLUE datasets with and without using a task-identifier. One reason for this can be that BERT captures the different dataset distributions such that the created sentence embeddings already encode which task / dataset the example is coming from and make sharing parameters in the model less useful. To test this hypothesis, I use the BERT-encoded datasets and train a logistic regression model to identify which dataset an example belongs to. This classifier reaches an accuracy of 97% on the datasets presented in Chapter 3. This result indicates that the BERT-encoded sentence representations already encode dataset-specific information which removes the need for a task-identifier and can also render standard multi-task learning less useful because the network is not forced to generalise between the different tasks. The large amount of task-specific information coming from BERT could also be linked to findings in using similar language models for unsupervised multi-task learning [29]. In a blog post by the authors of the paper, they write "On language tasks like question answering, reading comprehension, summarization, and translation, GPT-2 begins to learn these tasks from the raw text, using no task-specific training data. [...] The model is chameleon-like it adapts to the style and content of the conditioning text." [2]

To explore this finding further, I create two artificial datasets by splitting the largest

[2] https://openai.com/blog/better-language-models/

GLUE dataset, QQP - a paraphrase identification dataset, into two separate parts. For one of the halves, I reverse the labels such that the BERT encoded sentences come from the same distribution but the labels are from two separate (in this case opposite) tasks. As expected, the logistic regression model cannot distinguish between the two datasets and only reaches an accuracy of 50%.

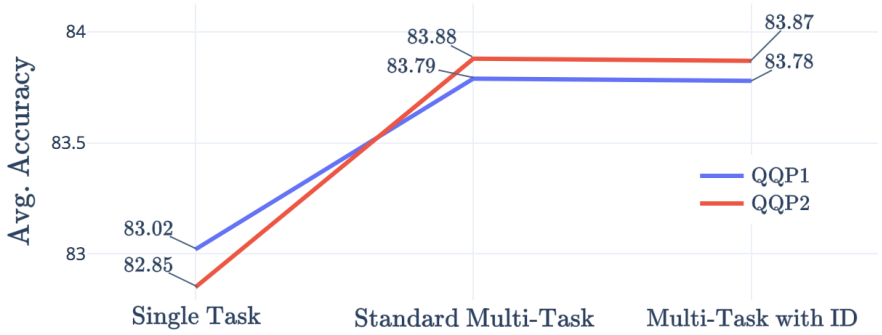

Figure 5.4: Performance improvement when using multi-task learning on the artificial QQP datasets. Average accuracy is computed over three independent runs.

With these two artificial datasets, I repeat the previously described multi-task learning experiments. Figure 5.4 shows the results for the single-task models, standard multi-task models and the multi-task models with an additional input task-identifier. The standard multi-task model improves over the single-task baselines with an average improvement of 0.76 and 1.02 in accuracy and standard deviation of 0.177 and 0.151 respectively. This result suggests that multi-tasking is more useful when the inputs of the different tasks are coming from a similar distribution. However, it is possible that simply reversing the labels for one half of the data is not a dissimilar enough task and the model shows an improvement because it has access to more data. Nevertheless, it can be argued that even though BERT sentence embeddings achieve impressive results as a fixed-size input to these models, their ability to encode the dataset origin of a training example makes multi-task learning less useful.

Despite the positive results with standard multi-task learning on these artificial datasets, multi-task learning with an additional task-identifier still does not significantly improve over the standard multi-task learning results. This suggests that the task-specific layer is enough to learn the task-specific output from the shared BERT representation. It remains unclear if the task identifier would be more useful with a less expressive input representation. To test this hypothesis, I compare the perfor-

mance of multi-task learning with BERT-encoded sentence representations against a simple bag-of-word (BOW) related approach. This BOW representation is achieved by creating a term frequency - inverse document frequency (tf-idf) vector for every sentence and then using truncated singular value decomposition to project the relatively high-dimensional sparse vector down to a smaller dimension of 128. This process is also known as latent semantic analysis (LSA) [20].

Unfortunately, this representation was not expressive enough to learn anything for most tasks from Chapter 3. It worked relatively well for QQP but I suspect that it posed problems for the harder task of textual entailment. For MRPC, which is also a paraphrase identification task like QQP the reason for this could be that the MRPC sentences are much longer on average. Since I cannot perform multi-task learning with only one task, I introduce two new tasks that only involve single sentence classification; SST-2 - a sentiment classification dataset and SUBJ - a subjectivity classification dataset. SST-2 contains about 67,000 sentences in the training set and SUBJ roughly 8,000. Both datasets are balanced. As in the previous experiments, it is interesting to see how much task-specific information can already be extracted from the input representation. Therefore, I train a logistic regression model to identify which task a sentence representation belongs to. With the BERT input representation the logistic regression model reaches an accuracy of 93% on these three tasks which is a bit lower than on the four originally chosen tasks. Compared to BERT, the BOW representation only gives an accuracy of 79% which is much smaller but still a relatively high accuracy overall. This suggests that BOW encodes less task-specific information than BERT which means that multi-task learning and a task-identifier may work better. However, it is also a sign that these three datasets are relatively easily distinguishable based on the text they are built on.

Figure 5.5 shows the improvement of multi-task trained models over the single-task baselines with the two different input representations for these datasets. From these results, it can be seen that using the task-identifier makes a much bigger difference for the BOW embeddings (yellow) than for the BERT embeddings (blue). For BERT, we see a very similar picture to all previous experiments. The single-task baselines are very hard to outperform. The standard deviation of the average improvement is 0.132 for QQP with the standard multi-task learning approach and 0.162 with the task-identifier. For, SST-2 the standard deviations are (in the same order) 0.392 and 0.23 and for SUBJ, they are 0.115 and 0.057. Taking into account the standard deviations, both multi-task models generally perform worse than the single-task baselines with

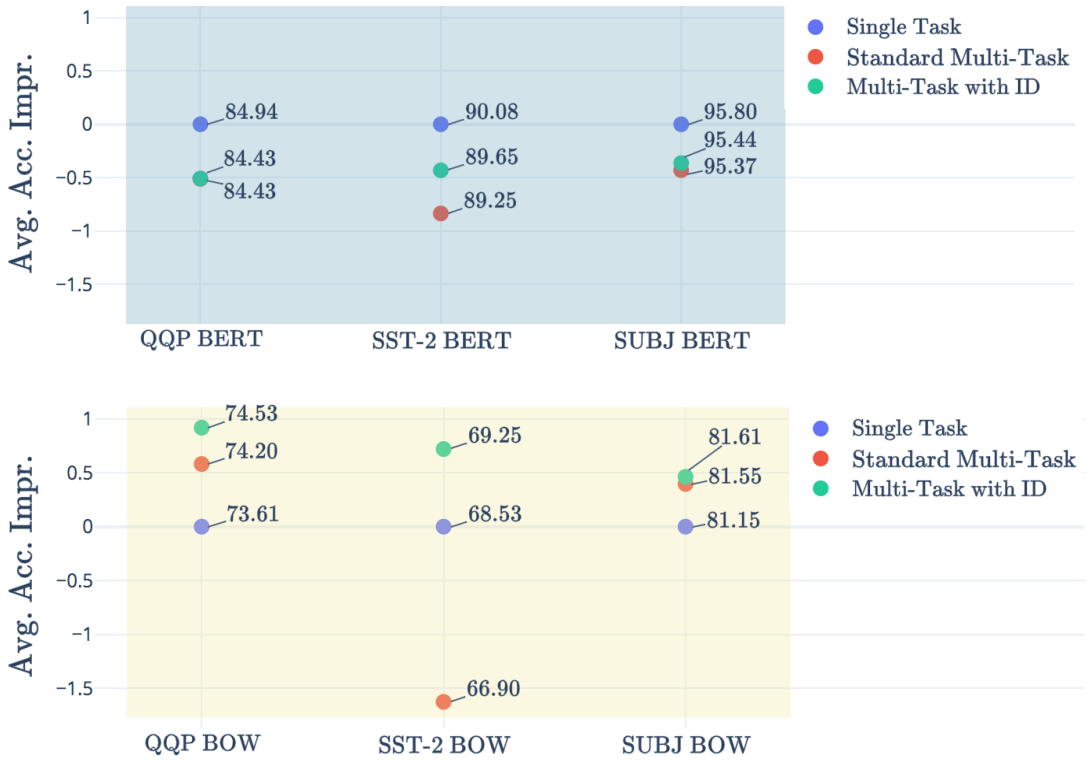

Figure 5.5: Influence of different input representations on improvement of multi-task learning approaches. The lower plot (yellow) shows the results of using a down-projected bag-of-words representation. The upper plot (blue) shows the results of using a BERT-encoded sentence representation.

BERT representations.

Interestingly, this is not the case for the experiments that used the BOW representation as the input representation (yellow). In contrast to the BERT representation, here, the multi-task models tend to outperform the single-task models. The standard deviations of the improvement are also smaller in most cases. For QQP, the standard deviations are 0.208 for standard multi-task learning and 0.119 with the task-id, for SST-2 0.086 and 0.199 and for SUBJ 0.099 and 0.057. The most striking result is the performance difference of the two multi-tasking experiments on SST-2. While multi-task learning with the task-identifier shows an improvement of 0.72 over the single-task baseline, the standard multi-task learning shows a drop in performance of -1.626 which is rather large considering the standard deviation of this loss is only 0.086. It is hard to say what causes this drop in accuracy on SST-2. One explanation might be that the shared layer learns to produce a representation which is beneficial for the two other

tasks but is less useful for the SST-2 task-specific layers.

For the model that has access to a task-identifier, there is an improvement for all the tasks compared to the single-task baselines. This result suggests that multi-task learning with a task-identifier can indeed be useful. In the standard multi-task experiment, the two other tasks, QQP and SUBJ, can profit from the shared representation but, proportionally, the loss on SST-2 is much bigger. Of course, this can still be a useful application for standard multi-task learning for example if QQP is the main task to improve and the two others are auxiliary tasks. In this case, the other tasks are meant to help QQP but the performance of the on the auxiliary tasks is not important. Nevertheless, this experiment has shown that it is possible to achieve an improvement on all involved tasks with the simple addition of a task-identifier.

It is important to keep in mind that this positive result was achieved using a much less expressive input representation than BERT. This BOW-based result on its own is not advancing the state-of-the-art. In fact, BERT is used for many state-of-the-art solutions at the moment. In Figure 5.5, it becomes clear that BERT really is a more powerful model since the raw accuracy of the BERT input model is much higher for every task compared to the BOW input. Therefore, in a real-life scenario it is most often not advisable to use a simpler input representation only to have a successful multi-task learning model. Even if the goal is to reduce the number of models used to solve a set of tasks, a multi-task model trained with BERT representations albeit not outperforming the single-task baselines still performs better than using a less expressive input representation. This can be very important in industrial settings where the focus is less on performance but more on efficiency.

Furthermore, this dissertation analysed the case of fixed-sized sentence representations. This was a sensible decision due to the short time frame and the limited available computing resources. However, other research with BERT either fine-tunes the whole BERT model which involves tuning many parameters or constructs token-based representations from a pretrained BERT model instead of whole sentence representations [8, 19]. As discussed previously, there is evidence that standard multi-task learning also does not outperform single-task baselines when BERT is fine-tuned [34, 6]. However, to draw clear conclusions on the merit of having a task-identifier as part of the input for multi-task learning, more experiments are needed. When fine-tuning BERT, the task-specific information can be included as an additional input token of the sequence instead of concatenating a sentence representation with a task-identifier embedding. This has the potential that the shared BERT model can use this task-specific informa-

tion to produce a "CLS" token representation that is more useful for the task-specific layers on top of BERT. Some preliminary experiments on fine-tuning BERT suggest that task-identifiers may be beneficial for low-resource tasks but more detailed research is needed.[3] Moreover, future experiments should also explore architectures that do not use pretrained contextual embedding methods and learn embeddings from scratch. For example, spaCy, a python library specialised in fast NLP solutions, explain that using BERT is too slow for their use case.[4] In such scenarios, it might be worth to further investigate multi-task learning models with task-identifiers.

Finally, in previous experiments with similar additional input tokens, for example in neural machine translation [13], all the parameters of the model are shared for all inputs and outputs. This is different from the discussed multi-task learning setup, where a task-specific layer is put on top of the shared layer. In their experiments, the additional identifier is essential to use a *single* model to translate between multiple languages. Similarly, as presented in the background section, McCann et al. [22] proposed a single model with no task-specific components that can solve numerous NLP tasks by rephrasing them as sequence-to-sequence question answering tasks. Again, this would not be possible without giving the network access to information about the current task it is supposed to solve. In both cases, it remains unclear whether a more standard multi-task learning setup with task-specific components would improve the performance of these models. Nevertheless, the simplicity of a single model in terms of number of parameters and its ability to do better zero-shot learning is often much more important than the performance [13] and additional input information such as task-identifiers play a central role to achieve this.

[3] Some details on this experiment are presented in the Appendix.

[4] `https://explosion.ai/blog/spacy-v2-1`

Chapter 6

Conclusions

This dissertation investigated the benefit of using an additional task-identifier that provides task-specific information for multi-task learning in NLP. My work built on previous findings that multi-task learning does not always work and its success depends largely on what tasks are involved. This motivated the idea that the model should have access to some information about what tasks it is solving to help it learn what to share between which tasks. Previous efforts in this direction focused on introducing new parameters that allow the multi-task model to learn which parts of the model to share [31, 16]. These approaches involve much bigger models in terms of parameters and are built on certain assumptions as to how the model should learn what to share. This dissertation introduced a much simpler approach that only requires an adaptation of the model as part of the input by giving the model access to a task-identifier.

Unfortunately, the multi-task learning results did not show an improvement over the single-task baselines. For standard multi-task learning, this is in line with other research which also reported that multi-task learning cannot outperform single-task models that involve fine-tuning BERT [34, 6]. In my initial experiments, there was also no evidence that the task-identifier improves multi-task learning. My hypothesis was that these results are due to the fixed-sized sentence representation obtained from a pretrained BERT model. I discussed this hypothesis in detail and presented findings that support this in Section 5.3. There is evidence that the BERT sentence representation encodes task-specific information which makes multi-task learning less useful since the input to the shared layers cannot be treated as task-independent. This also explains why there was not improvement when using a task-identifier; the model already received this information from the BERT-encoded sentence representation.

Moreover, I found that a less expressive input representation to the model gave

an improvement of multi-task learning over the single-task baselines. In these experiments, the additional task-identifier showed promising results for multi-task learning. However, this is not enough evidence to draw any general conclusions on its benefit. This means that I am unable to answer my two research questions with certainty. More experiments are needed that investigate the influence of task-identifiers when training embeddings from scratch and other approaches such as fine-tuning BERT or using ELMo or OpenAI GPT instead.

Future work on task-identifiers in multi-task learning should compare the results against other strategies that learn what parts of the network should be shared [30, 16]. This should involve an analysis of performance but also of model size and training time as I expect the latter two to be significantly lower with the approach presented in this dissertation. Multi-task learning with a task-identifier should also be investigated in other fields such as computer vision where the input representation is not necessarily reflecting the source domain as much as in NLP. Finally, it would be interesting to further investigate BERT-encoded sentence embeddings and explore in what cases it may be useful that they encode specific information about the origin of the encoded sentences. While it has not proven helpful for multi-task learning, it might have potential for other application areas such as improving information retrieval or text classification.

Bibliography

[1] Adrian Benton, Margaret Mitchell, and Dirk Hovy. Multi-task learning for mental health using social media text. *CoRR*, abs/1712.03538, 2017.

[2] Joachim Bingel and Anders 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*, pages 164–169, Valencia, Spain, April 2017. Association for Computational Linguistics.

[3] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 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.

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

[5] Chenhui Chu, Raj Dabre, and Sadao Kurohashi. An empirical comparison of domain adaptation methods for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 385–391, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[6] Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D. Manning, and Quoc V. Le. Bam! born-again multi-task networks for natural language understanding. In *ACL*, 2019.

[7] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[9] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[11] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *ICML*, 2019.

[12] L. C. Jain and L. R. Medsker. *Recurrent Neural Networks: Design and Applications*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1999.

[13] Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351, December 2017.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[15] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[16] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Deep multi-task learning with shared memory for text classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 118–127, Austin, Texas, November 2016. Association for Computational Linguistics.

[17] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth*

International Joint Conference on Artificial Intelligence, IJCAI'16, pages 2873–2879. AAAI Press, 2016.

[18] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 912–921, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[19] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*, 2019.

[20] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. Matrix decompositions and latent semantic indexing. *Introduction to Information Retrieval*, pages 403–417, 2008.

[21] Héctor Martínez Alonso and Barbara 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*, pages 44–53, Valencia, Spain, April 2017. Association for Computational Linguistics.

[22] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*, 2018.

[23] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[24] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003, 2016.

[25] Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. How transferable are neural networks in nlp applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 479–489, Austin, Texas, November 2016. Association for Computational Linguistics.

[26] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 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)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[27] Thai-Hoang Pham, Khai Mai, Nguyen Minh Trung, Nguyen Tuan Duc, Danushka Bolegala, Ryohei Sasano, and Satoshi Sekine. Multi-task learning with contextualized word representations for extented named entity recognition. *CoRR*, abs/1902.10118, 2019.

[28] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf*, 2018.

[29] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[30] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098, 2017.

[31] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Sluice networks: Learning what to share between loosely related tasks. *CoRR*, abs/1705.08142, 2017.

[32] Rico Sennrich, Barry Haddow, and Alexandra Birch. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 35–40, 2016.

[33] Anders Søgaard and Yoav Goldberg. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235, Berlin, Germany, August 2016. Association for Computational Linguistics.

[34] Asa Cooper Stickland and Iain Murray. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. In Kamalika Chaudhuri and

Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[35] Aarne Johannes Talman and Stergios Chatzikyriakidis. Testing the generalization power of neural network models across nli benchmarks. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 5 2019.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[37] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 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*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[38] Alex Wang, Ian F. Tenney, Yada Pruksachatkun, Katherin Yu, Jan Hula, Patrick Xia, Raghu Pappagari, Shuning Jin, R. Thomas McCoy, Roma Patel, Yinghui Huang, Jason Phang, Edouard Grave, Najoung Kim, Phu Mon Htut, Thibault F'evry, Berlin Chen, Nikita Nangia, Haokun Liu, , Anhad Mohananey, Shikha Bordia, Ellie Pavlick, and Samuel R. Bowman. jiant 1.0: A software toolkit for research on general-purpose text understanding models. `http://jiant.info/`, 2019.

[39] Adina Williams, Nikita Nangia, and Samuel 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)*, pages 1112–1122. Association for Computational Linguistics, 2018.

[40] Han Xiao. bert-as-service. `https://github.com/hanxiao/bert-as-service`, 2018.

[41] Kaisheng Yao, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu. Recurrent neural networks for language understanding. In *INTERSPEECH*, 2013.

Appendix A

Fine-Tuning BERT

This experiment is not a full-fledged analysis of fine-tuning BERT. Rather, it is a short experiment to check if task-identifiers may be beneficial for fine-tuning BERT in a multi-task learning setting. To perform the experiment, I used the framework developed by the people that run the GLUE benchmark [38]. The framework, jiant[1], provides a collection of configuration files and I used one that was designed for fine-tuning BERT in this experiment. I ensured that the same pretrained BERT model (bert-base-cased) was used as in the other experiments in this dissertation. Due to time constraints, I could only perform one run per experiment. I decided to test fine-tuning BERT on the four originally chosen datasets described in Chapter 3.

To use task-identifiers for the multi-task learning setup, I added a task-identifier as part of the input sequence at the beginning of every original sentence pair. This is possible in the case of fine-tuning because BERT can learn how to treat this task-identifier as part of the input sequence. This is in contrast to previous experiments in this dissertation on BERT-encoded sentence representations. In this case, the task-identifier information was passed through an embedding layer and concatenated with the sentence representation.

The results of the fine-tuning experiment can be seen in Table A.1. Following previous results with other experiments, the largest improvement with multi-task learning can be seen on RTE. In this case, there is also an improvement on MRPC, the second low-resource task. When using task-identifiers as part of the input sequence, the improvement on the two low-resource tasks increases while the performance on the two comparatively high-resource tasks, MNLI and QQP, drops further. Compared to the results with BERT-encoded sentence representations, fine-tuning achieves much higher

[1] https://github.com/nyu-mll/jiant

		Dev Acc.	Imp.	Dev F1	Imp.
STB	**MNLI**	78.70	-	-	-
	RTE	70.00	-	-	-
	QQP	87.00	-	82.80	-
	MRPC	86.30	-	90.40	-
SMT	**MNLI**	78.00	- 0.70	-	-
	RTE	71.50	+ 1.50	-	-
	QQP	86.00	- 0.30	82.10	- 0.70
	MRPC	87.30	+ 1.00	90.70	+ 0.30
MT-ID	**MNLI**	77.30	- 1.40	-	-
	RTE	72.90	+ 2.90	-	-
	QQP	85.10	- 1.90	81.10	- 1.70
	MRPC	88.70	+ 2.40	92.00	+ 1.60

Table A.1: Accuracy and F1-score of the single-task, standard multi-task and multi-task models with task-identifier trained by fine-tuning BERT. Results are computed over one independent run. Improvements are computed compared to the single-task baselines. STB = Single-Task Baseline, SMT = Standard Multi-Task Learning, MT-ID = Multi-Task Learning with Task-Identifier.

performance on all tasks. However, the obtained results still do not reach state-of-the-art results compared to other experiments on fine-tuning BERT that were uploaded to the GLUE evaluation server. This leads me to believe that the configuration file I took from jiant may have been designed for a toy experiment with fine-tuning rather than a competitive state-of-the-art setup. In any case, further experiments are needed to judge whether task-identifiers are beneficial when fine-tuning BERT in a multi-task learning scenario.

Appendix B

Tabular Results and Learning Curves

B.1 GLUE Benchmark Datasets

B.1.1 MNLI

	Loss	Accuracy	F1-Score
Train	0.257 ± 0.001	66.18 ± 0.2	66.20 ± 0.2
Dev	0.267 ± 0.002	64.00 ± 0.2	64.03 ± 0.1
Test	0.267 ± 0.000	63.78 ± 0.1	63.78 ± 0.1

Table B.1: Loss, accuracy and F1-score of the single-task model trained on the MNLI textual entailment dataset. Results are computed over three independent runs.

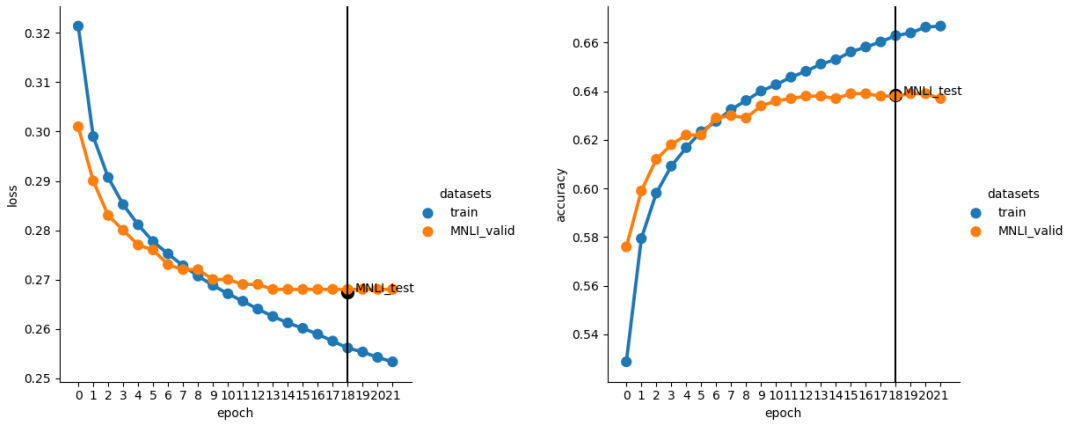

Figure B.1: Learning curve and accuracy of the single-task model trained on the MNLI textual entailment dataset. The black line marks the epoch where training was stopped and the black dot the loss / accuracy on the test set.

B.1.2 RTE

	Loss	Accuracy	F1-Score
Train	0.293 ± 0.002	69.56 ± 0.8	67.57 ± 0.4
Dev	0.322 ± 0.001	59.80 ± 1.3	57.53 ± 1.9
Test	0.328 ± 0.002	62.27 ± 0.5	59.26 ± 1.4

Table B.2: Loss, accuracy and F1-score of the single-task model trained on the RTE textual entailment dataset. Results are computed over three independent runs.

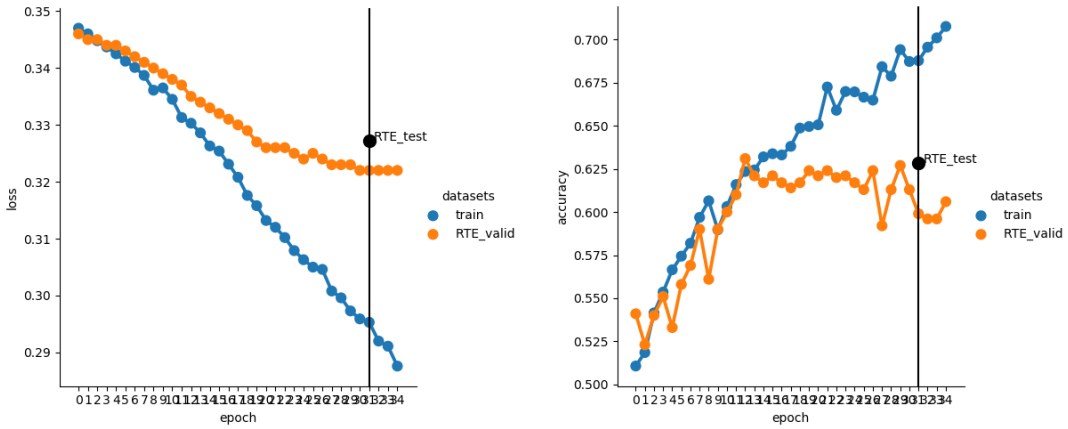

Figure B.2: Learning curve and accuracy of the single-task model trained on the RTE textual entailment dataset. The black line marks the epoch where training was stopped and the black dot the loss / accuracy on the test set.

B.1.3 QQP

	Loss	Accuracy	F1-Score
Train	0.146 ± 0.007	86.96 ± 0.8	82.35 ± 1.1
Dev	0.162 ± 0.001	84.97 ± 0.1	79.77 ± 0.4
Test	0.164 ± 0.001	84.94 ± 0.2	79.92 ± 0.4

Table B.3: Loss, accuracy and F1-score of the single-task model trained on the QQP paraphrase identification dataset. Results are computed over three independent runs.

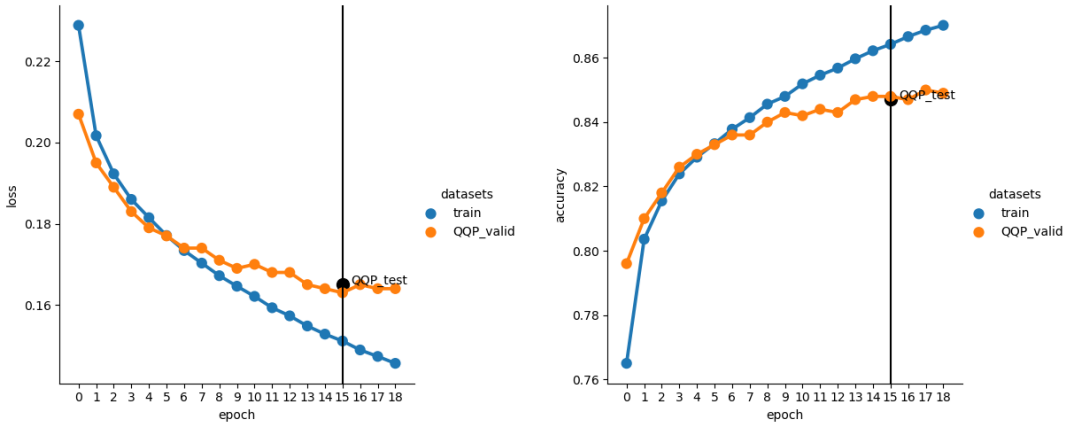

Figure B.3: Learning curve and accuracy of the single-task model trained on the QQP paraphrase identification dataset. The black line marks the epoch where training was stopped and the black dot the loss / accuracy on the test set.

B.1.4 MRPC

	Loss	Accuracy	F1-Score
Train	0.201 ± 0.009	81.38 ± 1.2	86.19 ± 0.8
Dev	0.237 ± 0.002	77.63 ± 0.6	84.37 ± 0.6
Test	0.230 ± 0.001	79.01 ± 0.7	85.26 ± 0.6

Table B.4: Loss, accuracy and F1-score of the single-task model trained on the MRPC paraphrase identification dataset. Results are computed over three independent runs.

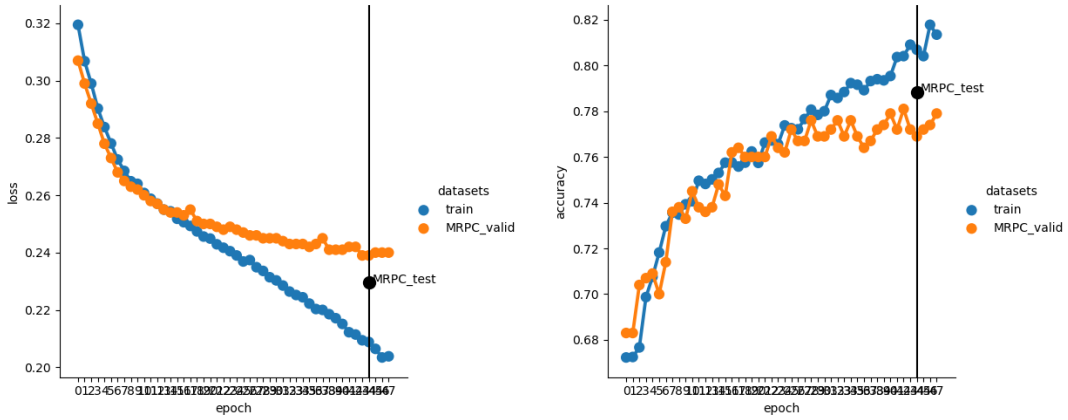

Figure B.4: Learning curve and accuracy of the single-task model trained on the MRPC paraphrase identification dataset. The black line marks the epoch where training was stopped and the black dot the loss / accuracy on the test set.

B.1.5 Standard Multi-Task Learning

	Test Loss	Imp.	Test Acc.	Imp.	Test F1	Imp.
MNLI	0.270 ± 0.001	+ 0.003	63.57 ± 0.1	- 0.21	63.54 ± 0.1	- 0.24
RTE	0.316 ± 0.006	- 0.012	64.00 ± 1.1	+ 1.73	64.14 ± 0.3	+ 4.88
QQP	0.165 ± 0.001	+ 0.001	84.72 ± 0.2	- 0.22	79.69 ± 0.2	- 0.23
MRPC	0.235 ± 0.003	+ 0.005	78.61 ± 0.6	- 0.40	85.17 ± 0.3	- 0.09

Table B.5: Loss, accuracy and F1-score of the standard multi-task model trained on the four chosen GLUE datasets. Results are computed over three independent runs. Improvements are computed compared to the baselines.

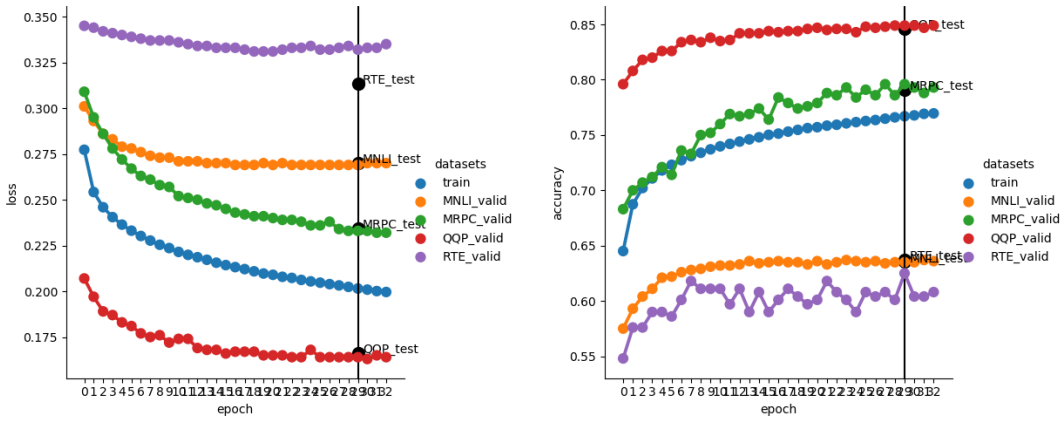

Figure B.5: Learning curve and accuracy of the standard multi-task model trained on the four chosen GLUE datasets. The black line marks the epoch where training was stopped and the black dot the loss / accuracy on the test set.

B.1.6 Multi-Task Learning with Task-Identifier

	Test Loss	Imp.	Test Acc.	Imp.	Test F1	Imp.
MNLI	0.269 ± 0.000	+ 0.002	63.59 + 0.1	- 0.19	63.59 ± 0.1	- 0.19
RTE	0.316 ± 0.002	- 0.012	63.87 ± 1.1	+ 1.60	64.01 ± 1.6	+ 4.75
QQP	0.165 ± 0.000	+ 0.001	84.89 ± 0.2	- 0.05	79.86 ± 0.4	- 0.06
MRPC	0.229 ± 0.003	- 0.001	78.69 ± 1.6	- 0.32	84.96 ± 1.0	- 0.30

Table B.6: Loss, accuracy and F1-score of the multi-task model with task-identifier trained on the four chosen GLUE datasets. Results are computed over three independent runs. Improvements are computed compared to the baselines.

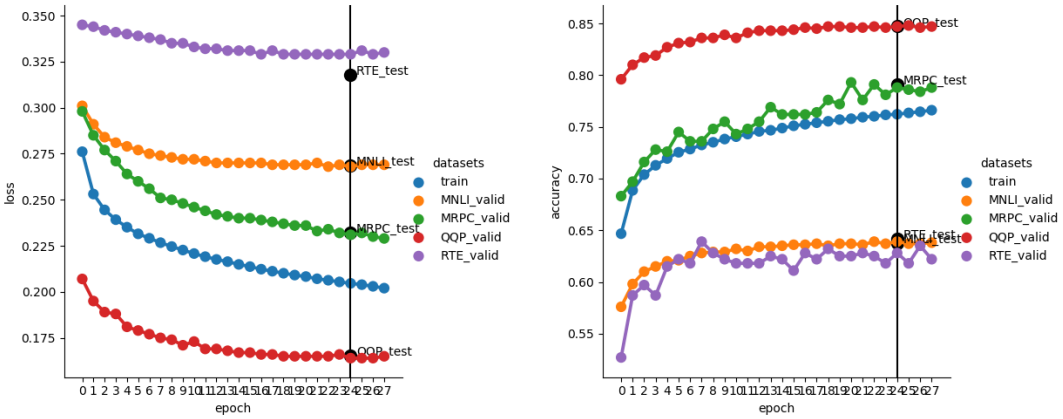

Figure B.6: Learning curve and accuracy of the multi-task model with task-identifier trained on the four chosen GLUE datasets. The black line marks the epoch where training was stopped and the black dot the loss / accuracy on the test set.

B.2 BERT-Related Experiments

B.2.1 QQP Artificial Dataset

		Test Loss	Imp.	Test Acc.	Imp.	Test F1	Imp.
STB	**QQP-1**	0.180 ± 0.001	-	83.02 ± 0.26	-	77.18 ± 0.46	-
	QQP-2	0.183 ± 0.001	-	82.85 ± 0.23	-	85.86 ± 0.17	-
SMT	**QQP-1**	0.172 ± 0.002	- 0.008	83.79 ± 0.17	+ 0.76	78.34 ± 0.14	+ 1.16
	QQP-2	0.174 ± 0.001	- 0.009	83.88 ± 0.36	+ 1.02	86.84 ± 0.30	+ 0.98
MT-ID	**QQP-1**	0.173 ± 0.001	- 0.006	83.78 ± 0.14	+ 0.75	78.34 ± 0.09	+ 1.16
	QQP-2	0.174 ± 0.000	- 0.009	83.87 ± 0.12	+ 1.02	86.93 ± 0.10	+ 1.07

Table B.7: Loss, accuracy and F1-score of the single-task, standard multi-task and multi-task models with task-identifier trained on the two artificial QQP datasets. Results are computed over three independent runs. Improvements are computed compared to the single-task baselines. STB = Single-Task Baseline, SMT = Standard Multi-Task Learning, MT-ID = Multi-Task Learning with Task-Identifier.

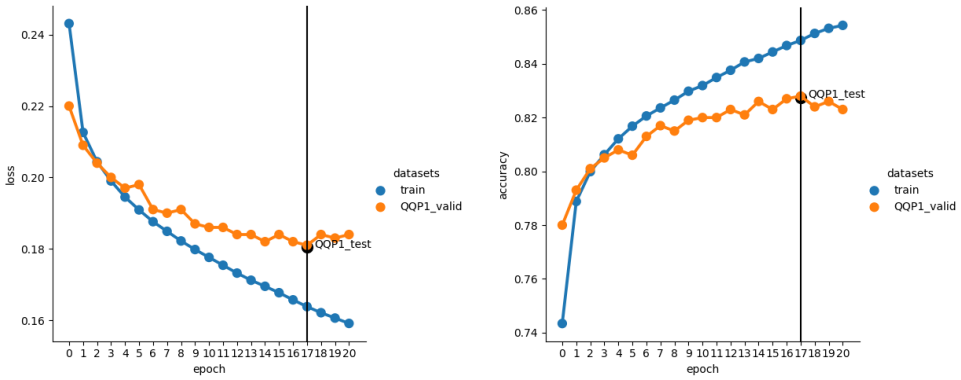

Figure B.7: Learning curve and accuracy of the single-task model trained on one half of the QQP dataset (where the labels are not reversed). The black line marks the epoch where training was stopped and the black dot the loss / accuracy on the test set.

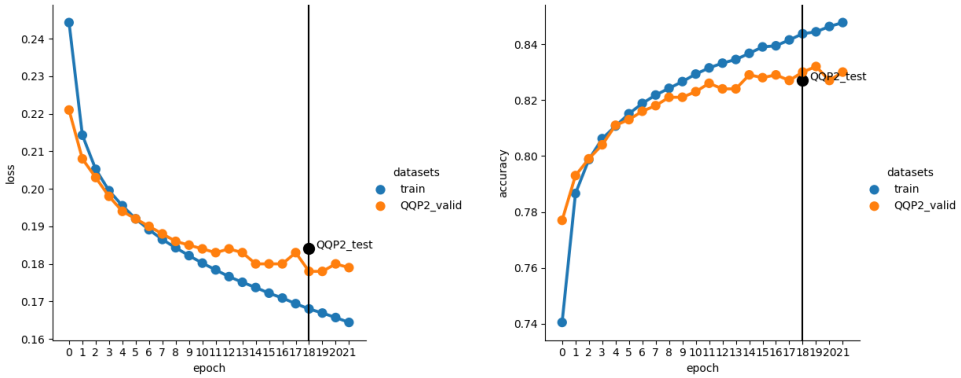

Figure B.8: Learning curve and accuracy of the single-task model trained on one half of the QQP dataset (where the labels are reversed). The black line marks the epoch where training was stopped and the black dot the loss / accuracy on the test set.

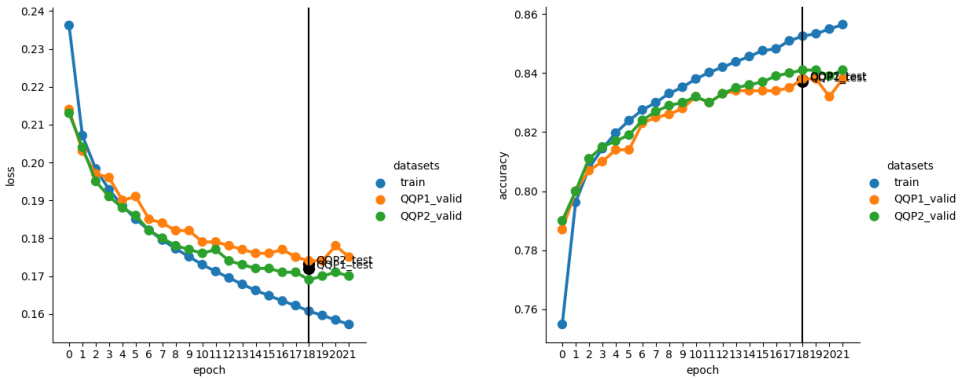

Figure B.9: Learning curve and accuracy of the standard multi-task model trained on the two artificial QQP datasets. The black line marks the epoch where training was stopped and the black dot the loss / accuracy on the test set.

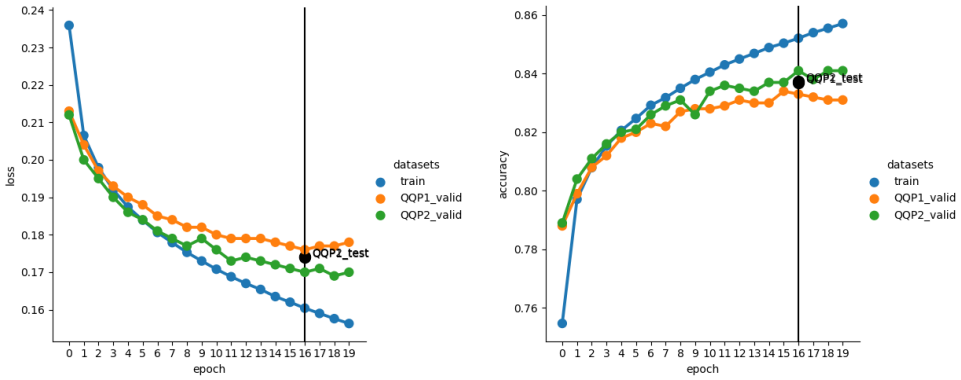

Figure B.10: Learning curve and accuracy of the multi-task model with task-identifier trained on the two artificial QQP datasets. The black line marks the epoch where training was stopped and the black dot the loss / accuracy on the test set.

B.2.2 BERT Input Representation

		Test Loss	Imp.	Test Acc.	Imp.	Test F1	Imp.
STB	**QQP**	0.164 ± 0.001	-	84.94 ± 0.22	-	79.92 ± 0.42	-
	SST-2	0.122 ± 0.001	-	90.08 ± 0.08	-	90.95 ± 0.04	-
	SUBJ	0.053 ± 0.001	-	95.80 ± 0.06	-	95.46 ± 0.05	-
SMT	**QQP**	0.169 ± 0.001	+ 0.005	84.43 ± 0.11	- 0.51	79.27 ± 0.12	- 0.65
	SST-2	0.126 ± 0.004	+ 0.004	89.25 ± 0.45	- 0.84	90.10 ± 0.46	- 0.84
	SUBJ	0.055 ± 0.001	+ 0.002	95.37 ± 0.15	- 0.43	94.96 ± 0.19	- 0.50
MT-ID	**QQP**	0.168 ± 0.001	+ 0.004	84.43 ± 0.06	- 0.51	79.08 ± 0.03	- 0.84
	SST-2	0.126 ± 0.002	+ 0.004	89.65 ± 0.27	- 0.43	90.48 ± 0.33	- 0.47
	SUBJ	0.055 ± 0.001	+ 0.002	95.44 ± 0.10	- 0.36	95.09 ± 0.15	- 0.37

Table B.8: Loss, accuracy and F1-score of the single-task, standard multi-task and multi-task models with task-identifier trained with BERT-encoded sentences as the input representation. Results are computed over three independent runs. Improvements are computed compared to the single-task baselines. STB = Single-Task Baseline, SMT = Standard Multi-Task Learning, MT-ID = Multi-Task Learning with Task-Identifier.

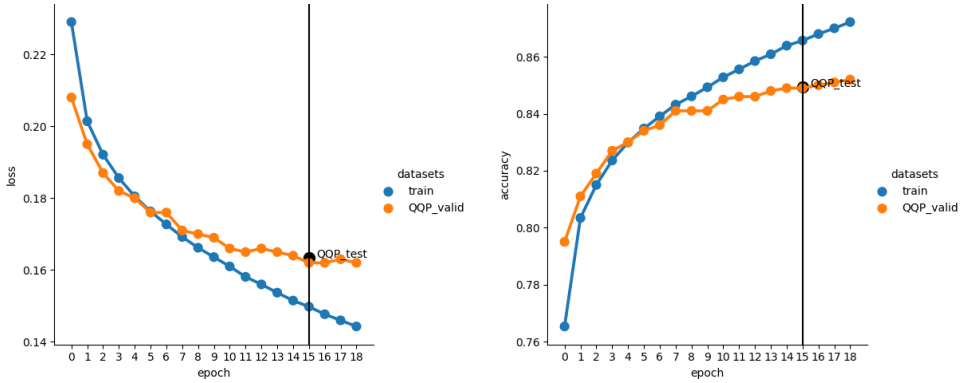

Figure B.11: Learning curve and accuracy of the single-task model trained on the QQP dataset. The black line marks the epoch where training was stopped and the black dot the loss / accuracy on the test set.

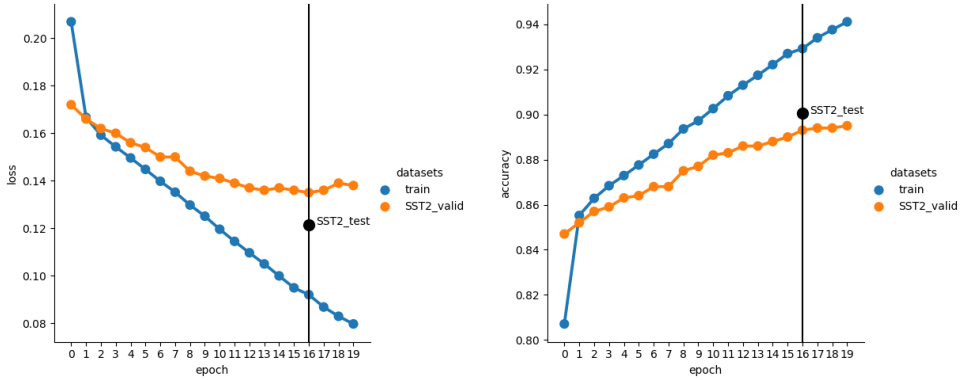

Figure B.12: Learning curve and accuracy of the single-task model trained on the SST-2 dataset. The black line marks the epoch where training was stopped and the black dot the loss / accuracy on the test set.

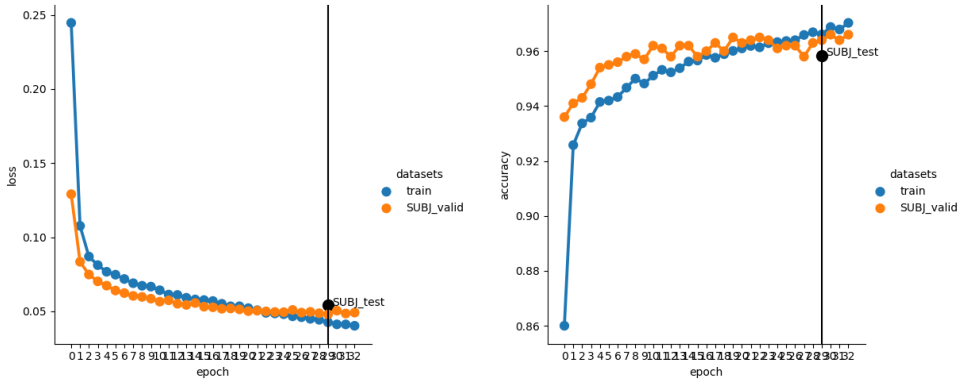

Figure B.13: Learning curve and accuracy of the single-task model trained on the SUBJ dataset. The black line marks the epoch where training was stopped and the black dot the loss / accuracy on the test set.

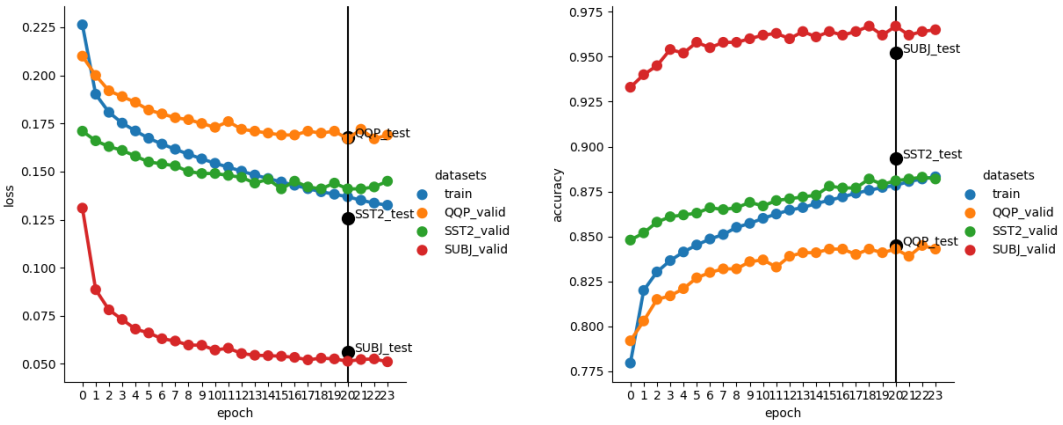

Figure B.14: Learning curve and accuracy of the standard multi-task model trained on the three datasets. The black line marks the epoch where training was stopped and the black dot the loss / accuracy on the test set.

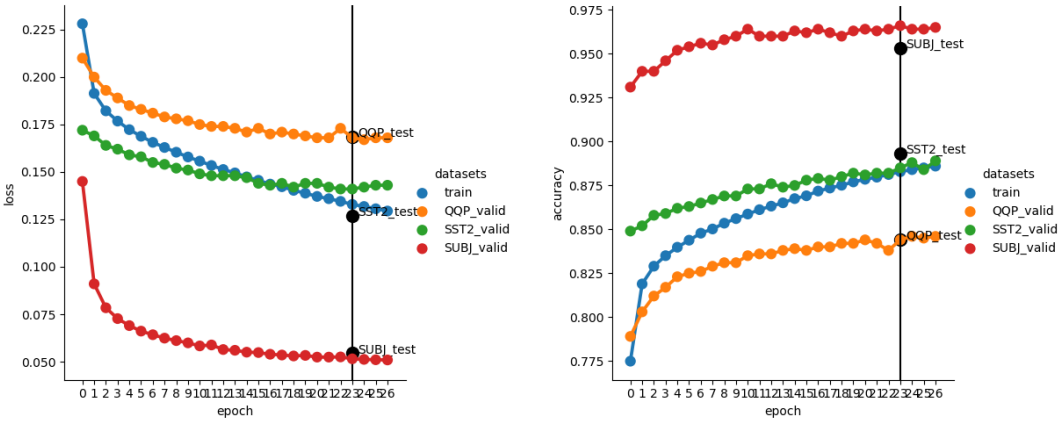

Figure B.15: Learning curve and accuracy of the multi-task model with task-identifier trained on three datasets. The black line marks the epoch where training was stopped and the black dot the loss / accuracy on the test set.

B.2.3 BOW Input Representation

		Test Loss	Imp.	Test Acc.	Imp.	Test F1	Imp.
STB	**QQP**	0.256 ± 0.002	-	73.61 ± 0.28	-	61.35 ± 0.47	-
	SST-2	0.287 ± 0.002	-	68.53 ± 0.32	-	74.91 ± 0.24	-
	SUBJ	0.210 ± 0.001	-	81.15 ± 0.10	-	80.45 ± 0.05	-
SMT	**QQP**	0.252 ± 0.000	- 0.004	74.20 ± 0.07	+ 0.58	62.06 ± 1.04	+ 0.72
	SST-2	0.293 ± 0.001	+ 0.006	66.90 ± 0.35	- 1.63	73.97 ± 0.37	- 0.94
	SUBJ	0.212 ± 0.000	+ 0.002	81.55 ± 0.17	+ 0.40	80.68 ± 0.12	+ 0.23
MT-ID	**QQP**	0.236 ± 0.024	- 0.020	74.53 ± 0.17	+ 0.92	62.65 ± 0.99	+ 1.30
	SST-2	0.283 ± 0.000	- 0.004	69.25 ± 0.13	+ 0.72	75.19 ± 0.24	+ 0.28
	SUBJ	0.208 ± 0.000	- 0.002	81.61 ± 0.15	+ 0.46	80.76 ± 0.09	+ 0.32

Table B.9: Loss, accuracy and F1-score of the single-task, standard multi-task and multi-task models with task-identifier trained with BOW-encoded sentences as the input representation. Results are computed over three independent runs. Improvements are computed compared to the single-task baselines. STB = Single-Task Baseline, SMT = Standard Multi-Task Learning, MT-ID = Multi-Task Learning with Task-Identifier.

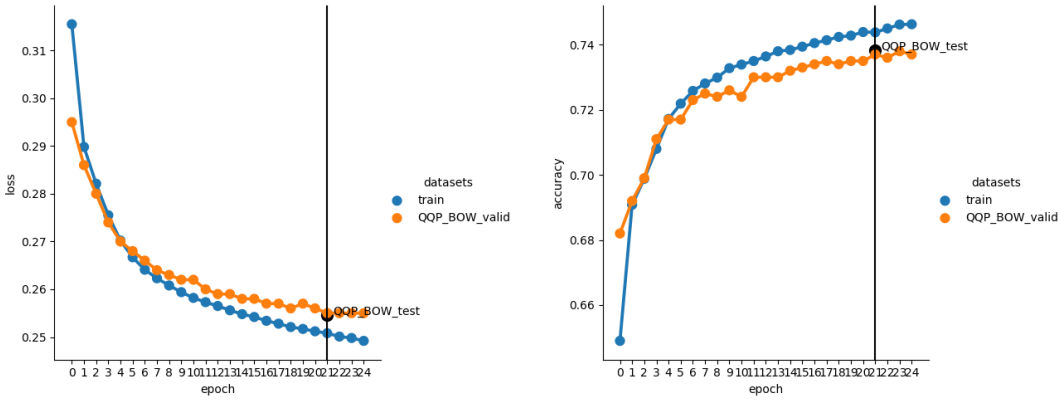

Figure B.16: Learning curve and accuracy of the single-task model trained on the QQP dataset. The black line marks the epoch where training was stopped and the black dot the loss / accuracy on the test set.

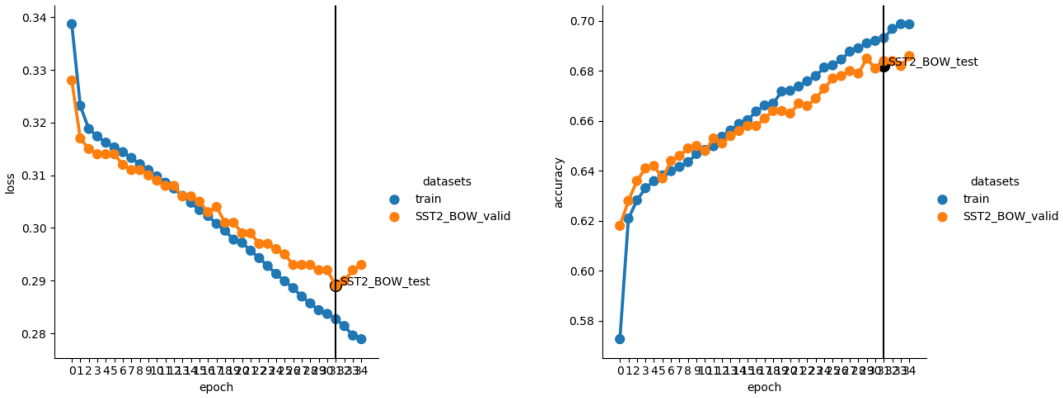

Figure B.17: Learning curve and accuracy of the single-task model trained on the SST-2 dataset. The black line marks the epoch where training was stopped and the black dot the loss / accuracy on the test set.

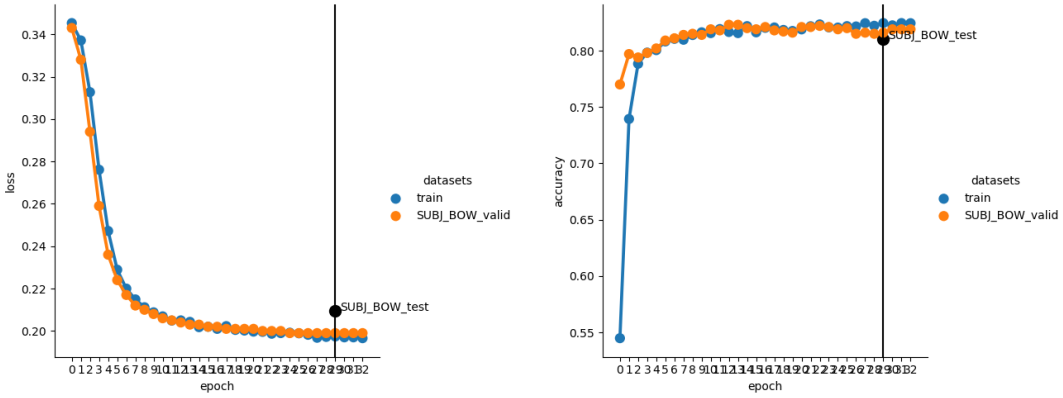

Figure B.18: Learning curve and accuracy of the single-task model trained on the SUBJ dataset. The black line marks the epoch where training was stopped and the black dot the loss / accuracy on the test set.

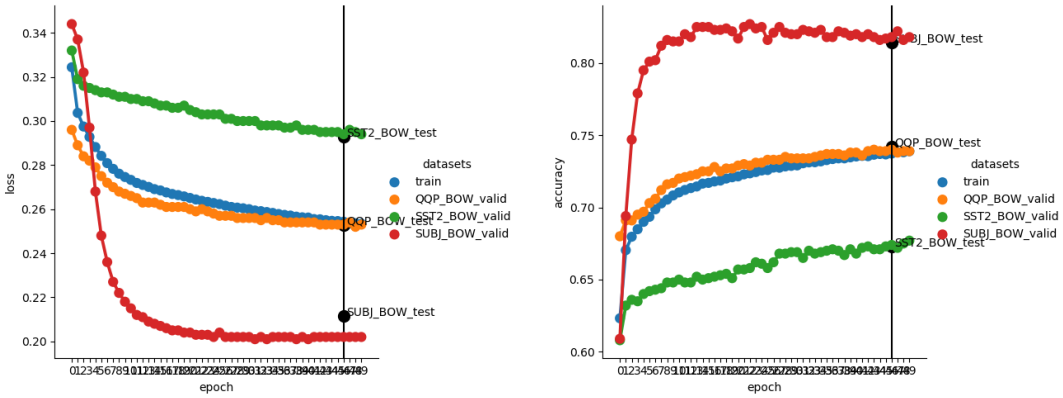

Figure B.19: Learning curve and accuracy of the standard multi-task model trained on the three datasets. The black line marks the epoch where training was stopped and the black dot the loss / accuracy on the test set.

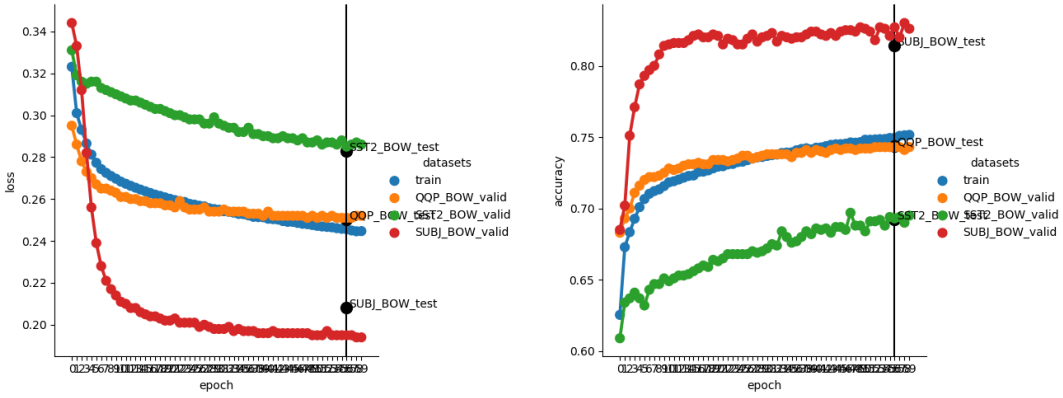

Figure B.20: Learning curve and accuracy of the multi-task model with task-identifier trained on three datasets. The black line marks the epoch where training was stopped and the black dot the loss / accuracy on the test set.