EasyTransfer: A Simple and Scalable Deep Transfer Learning Platform for NLP Applications

Minghui Qiu[1], Peng Li[1*], Chengyu Wang[1*], Haojie Pan[1*], Ang Wang[1], Cen Chen[1], Xianyan Jia[1], Yaliang Li[1], Jun Huang[1], Deng Cai[2], Wei Lin[1]
[1] Alibaba Group [2] State Key Lab of CAD&CG, Zhejiang University, China
{minghui.qmh,jerry.lp,chengyu.wcy,haojie.phj,chencen.cc,wangang.wa,xianyan.xianyanjia}@alibaba-inc.com
{yaliang.li,huangjun.hj,weilin.lw}@alibaba-inc.com,dengcai@cad.zju.edu.cn

ABSTRACT

The literature has witnessed the success of leveraging Pre-trained Language Models (PLMs) and Transfer Learning (TL) algorithms to a wide range of Natural Language Processing (NLP) applications, yet it is not easy to build an easy-to-use and scalable TL toolkit for this purpose. To bridge this gap, the EasyTransfer platform is designed to develop deep TL algorithms for NLP applications. EasyTransfer is backended with a high-performance and scalable engine for efficient training and inference, and also integrates comprehensive deep TL algorithms, to make the development of industrial-scale TL applications easier. In EasyTransfer, the built-in data and model parallelism strategies, combined with AI compiler optimization, show to be 4.0x faster than the community version of distributed training. EasyTransfer supports various NLP models in the ModelZoo, including mainstream PLMs and multi-modality models. It also features various in-house developed TL algorithms, together with the AppZoo for NLP applications. The toolkit is convenient for users to quickly start model training, evaluation, and online deployment. EasyTransfer is currently deployed at Alibaba to support a variety of business scenarios, including item recommendation, personalized search, conversational question answering, etc. Extensive experiments on real-world datasets and online applications show that EasyTransfer is suitable for online production with cutting-edge performance for various applications. The source code of EasyTransfer is released at Github [1].

CCS CONCEPTS

• **Computing methodologies → Neural networks**.

KEYWORDS

Natural Language Processing, Pre-trained Language Model, Transfer Learning

[*] Equal contributions.
[1] https://github.com/alibaba/EasyTransfer

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3481911

ACM Reference Format:
M. Qiu, P. Li, C. Wang, H. Pan, A. Wang, C. Chen, X. Jia, Y. Li, J. Huang, D. Cai, W. Lin. 2021. EasyTransfer: A Simple and Scalable Deep Transfer Learning Platform for NLP Applications. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3481911

1 INTRODUCTION

Transfer Learning (TL) is a rapidly growing field of machine learning that aims to improve the learning of a data-deficient task by transferring knowledge from related data-sufficient tasks [27, 42, 54]. Witnessing the great representation learning abilities of deep neural networks, neural TL methods, i.e., deep transfer learning, have gained increasing popularity and are shown to be effective for a wide variety of applications [25, 37, 49, 53]. Despite the success of deep TL methods for real-world applications, it is still very challenging to design a deep TL framework. There are a few earliest attempts to build TL toolkits. Notable projects include:

- The NVIDIA Transfer Learning Toolkit (TLT) [2] is a python-based toolkit for training models and customizing them with users' own datasets. It mainly focuses on computer vision.
- Amazon Xfer [3] is an MXNet library that largely automates deep TL. It contains the "ModelHandler" component to extract features from pre-trained models and the "Repurposer" component to re-purpose models for target tasks.
- Transfer Learning Toolkit [4] integrates five types of models, namely feature-based, concept-based, parameter-based, instance-based, and deep-learning-based.
- The Huggingface Transformers toolkit [5] specifically addresses model-finetuning, especially for BERT-like models. It is backended by PyTorch and Tensorflow 2.0 and integrates Pre-trained Language Models (PLMs).

Challenges. However, when it comes to industrial-scale real-world applications, the above-mentioned toolkits might be less ideal. The reasons are threefold. i) PLMs such as BERT [6], T5 [33] and GPT [31] are becoming larger and usually have parameters on the billion scale. To elaborate, T5-large [33] and GPT-3 are with 11 and 175 billion parameters respectively, which exceed the memory limit of modern processors. It is needed to build a toolkit to support scalable distributed training strategies to train and deploy such

[2] https://developer.nvidia.com/transfer-learning-toolkit
[3] https://github.com/amzn/xfer
[4] https://github.com/FuzhenZhuang/Transfer-Learning-Toolkit
[5] https://github.com/huggingface/transformers

Table 1: An overview of transfer learning toolkits.

Framework	ModelZoo	AppZoo	TL Algorithms					Production-ready
			Fine-tuning	Feature-based	Instance-based	Model-based	Meta Learning	
NVIDIA TLT	✓		✓					
Amazon Xfer	✓		✓					
Hugginngface Transformers	✓		✓			✓		
TL Toolkit	✓		✓	✓	✓	✓		
EasyTransfer	✓	✓	✓	✓	✓	✓	✓	✓

models efficiently and effectively in real-world applications. ii) Real-world applications with diverse data properties require different types of TL algorithms, yet there are no TL toolkits available for users to examine rich types of state-of-the-art TL algorithms. There is an increasing need for a toolkit that supports a comprehensive suite of TL algorithms. iii) A gap still exists between developing an algorithm for a specific task and deploying the algorithm for online production. For many online applications, it is still a non-trivial task to provide a reliable service with high QPS (Queries Per Second) requirements. In a nutshell, it is highly necessary to develop a comprehensive, industry-scale deep TL toolkit.

Our Work. In light of these challenges, we develop the EasyTransfer toolkit and release it to the open-source community. EasyTransfer is built with highly scalable distributed training strategies, facilitating super large-scale model training. It supports a comprehensive suite of TL algorithms that can be used in various NLP tasks, providing a unified framework of model training, inference, and deployment for real-world applications. Currently, we have integrated EasyTransfer into a number of deep learning products in Alibaba and observed notable performance gains. Table 1 summarizes the main differences between EasyTransfer and existing TL toolkits. Comparing to other toolkits, EasyTransfer provides more comprehensive and scalable TL algorithms and functionalities for developing and deploying NLP applications. In summary, this paper makes the following contributions:

- We are the first to propose a simple and scalable deep TL platform named EasyTransfer to make it easy to develop deep TL algorithms for NLP applications. It is built with efficient distributed training strategies and AI compiler optimization for training super large-scale NLP models;
- EasyTransfer provides a rich family of TL algorithms to cover a wide range of applications in industry, including state-of-the-art and in-house developed algorithms where all these algorithms are shown to be effective in real-world applications;
- EasyTransfer is equipped with ModelZoo, containing more than 20 mainstream PLMs and a multi-modality model. EasyTransfer further integrates AppZoo and simple user interfaces to support for building NLP applications
- Extensive experiments have demonstrated the efficiency and scalability of the toolkit, and also the effectiveness in real-world applications. The toolkit has been deployed at Alibaba to support a variety of business scenarios (more than 20+ applications). EasyTransfer is seamlessly integrated with Platform of AI (PAI) products [6], to make it easy for external users outside of Alibaba to conduct model training, evaluation and online deployment on the cloud.

EasyTransfer is released under the Apache 2.0 License and open-sourced at GitHub. The detailed documentation and tutorials are available on this link [7]. The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 and Section 4 present EasyTransfer infrastructure and algorithms. Extensive experiments in Section 5 examine the efficiency, scalability and effectiveness of EasyTransfer. The last section draws the conclusion.

2 RELATED WORK

In this section, we summarize the related work on transformer-based PLMs and various deep TL methods.

2.1 Transformer-based PLMs

Transformers. For PLMs, transformers can primarily be applied in three ways: encoder-only, decoder-only, and encoder-decoder. The original transformer [44] was designed as a machine translation architecture. BART [17] and T5 [33] leverage this architecture to achieve state-of-the-art performance on downstream classification tasks. Previous work leveraging transformers for language modeling such as BERT [6], RoBERTa [22]and ALBERT [15] only use the encoder architecture. GPT-2 [32] only uses the decoder architecture to solve text generation tasks. Recent studies also find transformer architectures are effective for modeling cross-modality datasets, e.g. ViLBERT [23], Unicoder-VL [19] and VLBERT [40] for image and text data. EasyTransfer features FashionBERT [7], which studies cross-modality learning with transformers for e-commerce.

Sparse Transformers. The complexity of self-attention in transformers grows quadratically with the sequence length. Recent studies consider sparse transformers to alleviate this problem, which mainly fall into three groups: pattern-based, memory-based and low-rank. Pattern-based methods blockify attention matrices by limiting the field of view. BlockBERT [29] chunks input sequences into fixed blocks. Sparse Transformer [5] combines strided and local attention by assigning half of its heads to the pattern. Memory-based methods such as Longformer [1] use global memory to access the entire sequence. Set Transformers [16] applies inducing points from sparse Gaussian process to reduce the complexity. Low-rank approximation seeks to build a low-rank representation of full-attention matrices. Linformer [47] projects the dimension length

[6]https://www.alibabacloud.com/product/machine-learning
[7]https://www.yuque.com/easytransfer/cn

of keys and values to a lower dimension. EasyTransfer develops a novel sparse attention kernel that reduces memory footprints while preserving similar downstream task performance.

2.2 Deep TL Methods

In the literature, different types of TL models have been proposed over the years, depending on how the knowledge is shared [27, 42]. In this work, we categorize deep TL models into five types, i.e., *model fine-tuning*, *feature-based*, *instance-based*, *model-based*, and *meta learning*, detailed in the following.

Model fine-tuning. It is a simple but effective TL paradigm, where the pre-trained model in the source domain is used as the initialization for continued training on the target domain data. Model fine-tuning has been broadly used to reduce the number of labeled data needed for learning new tasks and tasks in new domains [6, 8, 11].

Feature-based methods. This type of methods aims to locate a common feature space that can reduce the differences between the source and target domains. They transform features from one domain to be closer to another, or project features of different domains into a common latent space where the feature distributions are close to each other [39]. With the recent advances of deep learning, different NN-based TL frameworks have been proposed for feature-based TL [21, 24, 25, 39, 50]. A simple but widely used framework is to train a shared NN to learn a shared feature space [25]. Another representative framework is to employ a shared NN and two domain-specific NNs to respectively derive a shared feature space and two domain-specific feature spaces [21]. Adversarial training is also adopted to help learn better feature representations [21]. EasyTransfer features Domain-Relationship Specific Shared (DRSS) method to jointly learn the shared feature representations and domain relationships in a unified model.

Instance-based methods. This type of methods seeks to re-weight the source samples so that data from the source domain and the target domain would share a similar data distribution [4, 12, 30, 35, 45]. The TL module is typically considered as a sub-module of the data selection framework [35]. The studies in [30, 45] consider building a reinforced selector to help select high-quality source data to help the target.

Model-based methods. These methods try to transfer the knowledge from the model itself and usually are applied by distilling a big model to a small model [3, 9, 34]. Recently, the research of knowledge distillation (KD) from a large pretrain language model becomes popular. Tang et al. [43] distills BERT into BiLSTM networks and achieve comparable results. DistilBERT [36] applies KD loss in the pre-training stage, while BERT-PKD [41] distills BERT into shallow Transformers in the fine-tuning stage. TinyBERT [13] further applies the distillation of hidden attention matrices and embedding matrices and distills BERT with a two-stage KD process. AdaBERT [3] uses neural architecture search to adaptively find small architectures for different tasks. EasyTransfer develops the MetaKD method to use meta-teacher learning and meta-distillation to digest transferable knowledge across domains while knowledge distillation and achieves better results on cross-domain tasks.

Meta learning. This type of methods is slightly different from previous TL categories as meta learning does not directly transfer knowledge from source domains to target ones. Instead, it aims at learning "meta-learners" that can digest knowledge from various tasks [10, 28]. After the training of "meta-learners", only a few fine-tuning steps are sufficient for "meta-learners" to adapt to these tasks with high performance. In EasyTransfer, we treat meta learning as a special type of TL method and design cross-domain and cross-task training algorithms to capture the transferable knowledge.

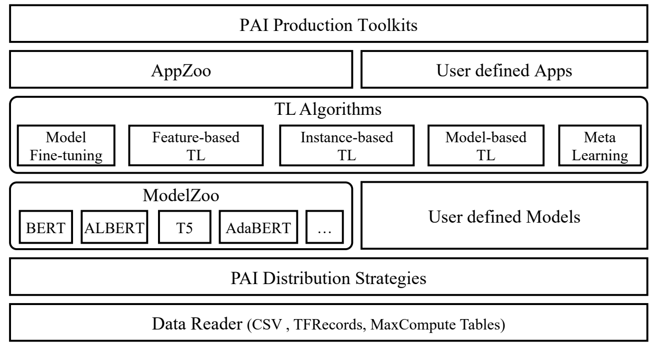

Figure 1: An overview of the EasyTransfer toolkit.

3 EASYTRANSFER INFRASTRUCTURE

In this section, we provide an overview of the EasyTransfer toolkit. The high-level framework of EasyTransfer is shown in Figure 1. EasyTransfer supports rich data readers to process data from multiple data sources and integrates the efficient distribution strategies from Alibaba Platform of AI (PAI). Based on this, users can either the pre-trained models from the ModelZoo or use EasyTransfer APIs to build their own models. EasyTransfer supports the mainstream TL algorithms and a set of in-house developed TL algorithms. It is also equipped with the PAI production toolkit to make it easy for online deployment.

3.1 Data Storage

Besides data storage in TFRecord or in the vanilla CSV format, EasyTransfer recommends storing data in MaxCompute [8] tables for high-performance data reading and caching to improve model training efficiency. The MaxCompute platform has server layers to split a data reading job into multiple sub-jobs automatically to support data sharding for distributed training and prediction. Details of resource scheduling in MaxpCompute can be found in [51].

3.2 Distributed Training Strategies

EasyTransfer leverages Whale, a distributed training framework developed by PAI, that facilitates the distributed training of industrial-level giant models. Whale provides multiple parallelism strategies including Data Parallelism (DP), Pipeline and a hybrid strategy that combines both DP and Pipeline.

Whale DP. The key difference between Whale DP and standard DP lies in the optimization of communication burden. We find that both intra-machine (between multiple local GPUs) and inter-machine (between workers) communications account for a high proportion

[8]https://www.alibabacloud.com/product/maxcompute

of running time, especially for large models. As the intra-machine communication bandwidth is much higher than inter-machine bandwidth, inter-machine communication becomes a bottleneck for high-speed training. To alleviate this, we consider hierarchical collective communication to first perform intra-machine data aggregation and then inter-machine aggregation. This helps to reduce the inter-machine communication burden. Furthermore, we consider fusing gradients of different tensors together to reduce communication overhead. We first estimate the generation time of different tensors in the static model graph and then fuse those tensors with similar generation time together for communication. Last but not least, Whale also supports deep gradient compression [20] and half-precision compression.

Whale HP. When it comes to super large model training where the model cannot be hosted in a single GPU, the above DP strategy is not sufficient. To support this, Whale considers a Hybrid Parallelism (HP) strategy to combine DP with model parallelism and Pipeline. The key idea in model parallelism is to partition large model nicely to different devices so that each device has a workload up to its own memory and computation capability. Since a model is split into different devices, there is a dependency between devices when performing feed-forward and backward operations. Pipeline is further used to reduce the waiting time between devices in these scenarios. Whale considers an autograph partition module to partition a model according to the Floating-Point Operations Per Second (FLOPS) and memory cost of each stage of model training. By cooperating with auto gradient checkpointing, Whale could balance the FLOPs and memory cost among pipeline stages on different devices and improve the overall training efficiencies for large models. By default, we consider low-precision weights (FP16) for communication to achieve better training efficiency.

By using Whale, EasyTransfer serves users with more efficient and powerful distributed training capabilities. EasyTransfer is also equipped with our AI Compiler optimizations, it is observed that memory access volume and framework overhead can be saved significantly, details can be found in [52].

3.3 ModelZoo & AppZoo

To help users better develop NLP models and applications, EasyTransfer provides a rich family of pretrained models in ModelZoo and a comprehensive NLP applications in AppZoo.

ModelZoo. PLMs such as BERT [6], RoBERTa [22], T5 [33] and GPT2 [32] have been the most successful deep learning models for NLP. To help users to train models with PLMs, we build ModelZoo in EasyTransfer, which offers pre-trained models including mainstream PLMs including all mainstream transformer-based PLMs, together with the cross-modality model FashionBERT [7]. Our ModelZoo is fully compatible with pre-trained models from open-source toolkits such as Huggingface Transformers.

Table 2 presents pre-trained models that EasyTransfer offers to the public with different model sizes ranging from tiny to xxLarge. Note that, Base models have the hidden size of 768, 12 layers, and 12 attention heads, Large models have the hidden size of 1024, 24 layers, and 16 attention heads, xLarge models have the hidden size of 2048, 24 layers, and 32 attention heads, and xxLarge models have the hidden size of 4096, 12 layers and 64 attention heads.

Table 2: The EasyTransfer ModelZoo. cn means models pre-trained over the Chinese corpus. en means models pre-trained over the English corpus.

ModelZoo	Tiny	Base	Large	xLarge	xxLarge
BERT-cn/en	✓	✓	✓		
ALBERT-cn/en	✓	✓	✓	✓	✓
RoBERTa-cn/en	✓	✓	✓		
T5-cn/en	✓	✓	✓		

We will also release our pre-trained super-large scale models (with up to 100 billion parameters) in the near future.

Random Sparse Attention (RSA). The key innovation in transformer-based PLMs is the self-attention mechanism, which is highly effective in capturing the relationship between tokens in an input sequence. However, the attention computation and memory consumption grow quadratically with the sequence length n, which limits their application to real-world scenarios that require long sequence modeling such as document modeling and cross-modality learning. To address this limitation, we develop a novel sparse attention kernel that reduces memory footprints of attention computation via random block-based sparse multi-head attention. More specifically, our approach first blockifies the attention matrices, randomly selects some blocks and then distributes the blocks over multiple attention heads. Our random sparse attention decreases the memory footprints and computation FLOPs from $O(n^2)$ to $O(n^2/k)$ (where k is the number of the local attention blocks), and keeps the model quality comparable with the full-attention pre-trained models.

AppZoo. To help users better develop NLP applications with our toolkit, we further provide a comprehensive NLP application tool AppZoo. It supports running applications with a few command-line arguments and provides 10+ mainstream NLP application models for users. The AppZoo provides rich modules for users to build different application pipelines, supporting four classes of NLP applications, including text classification, text matching, machine reading comprehension, and sequence labeling, with more than 10 models. The detailed application list is at Table 4.

3.4 Production Ready Platform

One advantage of the EasyTransfer platform is that it can directly generate production-ready models for efficient online prediction. After a training procedure is completed, a serialization model in the SavedModel format can be automatically generated from the checkpoint with the highest validation performance. As a default setting, EasyTransfer allows users to deploy the models on PAI Elastic Algorithm Service (EAS), which is an online prediction service that supports deploying machine learning models as RESTful APIs based on heterogeneous hardware (either CPUs or GPUs). By sending simple HTTP requests, users can use the online model service with high concurrency and stability. It is worth mentioning that users are free to use other online prediction services as well.

Table 3: TL algorithms supported by EasyTransfer. Algorithms marked as "new" are in-house developed methods.

Type	Algorithms	New
Model Fine-tuning	ModelZoo (BERT, GPT, etc.)	
	FashionBERT [7]	✓
Feature-based TL	FullyShared [25]	
	SharedPrivate [21]	
	SharedPrivate-Adv [21]	
	DRSS [50]	✓
	DRSS-Adv [50]	✓
Instance-based TL	Ruder & Plank [35]	
	RTL [30]	✓
	MGTL [45]	✓
Model-based TL	BERT-PKD [41]	
	DistillBERT [36]	
	TinyBERT [13]	
	AdaBERT [3]	✓
	MetaKD [26]	✓
Meta Learning	MetaFT [46]	✓
	MetaDTL	✓

Table 4: The model list of the EasyTransfer AppZoo. Here, "ModelZoo" means that one can use any PLMs from ModelZoo such as BERT, GPT, etc. for fine-tuning on downstreaming tasks.

Application Type	Model
Text Classification	ModelZoo TextCNN
Text Matching	ModelZoo BiCNN, HCNN DAM, DAM+
Machine Reading Comprehension	ModelZoo HAE
Sequence Labeling	ModelZoo BERT-CRF BERT-BiLSTM-CRF

4 EASYTRANSFER ALGORITHMS

Recall that, the diverse real-world applications require different types of TL algorithms, yet there are no TL toolkits available for users to examine the state-of-the-art TL algorithms and also develop new TL algorithms. To bridge this gap, EasyTransfer not only integrates the main-stream TL algorithms but also provides a comprehensive suite of in-house developed TL algorithms for users to explore. Table 3 provides an overview of algorithms supported in EasyTransfer (those algorithms marked as "new" in the table are newly developed methods). With EasyTransfer, users can either develop their own algorithms using the build-in APIs or directly use the in-house developed algorithms.

Code 1: Model fine-tuning example.

```
class TextClassification(base_model):

    def build_logits(self, features, mode=None):
        model = model_zoo.get_pretrained_model('bert_base_en')
        dense = layers.Dense(self.num_labels)
        _, output = model([features], mode=mode)
        return dense(output), label_ids

    def build_loss(self, logits, labels):
        return softmax_cross_entropy(labels, self.num_labels, logits)

    def build_eval_metrics(self, logits, labels):
        return classification_eval_metrics(logits, labels, self.num_labels)
```

4.1 À la Carte Algorithms

Based on the above infrastructure, EasyTransfer provides both low-level and high-level layer APIs for users to build their own algorithms (à la carte). The layers include basic deep learning layers such as *dense*, *linear* and *LSTM*, NLP layers such as *BERT* and *Transformer*, and Convolution (CV) layers such as *Conv* and *Flatten*. These layers can be also combined with standard layers in Tensorflow[9]. Users can use pre-trained models from ModelZoo to build their applications. For instance, "bert_base_en" is one of the pre-trained models in ModelZoo (i.e., the BERT-base model for English). The detailed list of pre-trained models that we provide can be found in the documentation [10]. Furthermore, users can use pre-defined layers or standard Tensorflow layers to build their models. For example, Code 1 shows an example of building a BERT-based fine-tuning model for text classification.

4.2 In-house Developed Algorithms

EasyTransfer covers TL algorithms in all the five directions of TL namely model fine-tuning, feature-based TL, instance-based TL, model-based TL, and meta learning. All these algorithms have been tested in real-world applications and deployed online to support real-world business.

Model Fine-tuning. The most widely used TL algorithm for PLMs is model fine-tuning. For example, a few fine-tuning steps on BERT and T5 can achieve remarkable results for many NLP applications[6]. We have also provided a wide range of language models pre-trained using our collected datasets and based on the PAI platform. As for the cross-modality model, we have developed the FashionBERT model [7] for the fashion domain. It has been deployed in Alibaba and has proved highly effective in the tasks of item representation learning and item search.

Feature-based TL. These methods seek to locate a common feature space that can reduce the differences between source and target domains, by transforming the features from one domain to be closer to another, or projecting different domains into a common latent space where the feature distributions are close [39]. Besides these algorithms, EasyTransfer features the Domain-Relationship Specific Shared (DRSS) algorithm to simultaneously learn shared representations and domain relationships in a unified framework [50]. DRSS can capture the inter-domain and intra-domain relationships within

[9]https://github.com/tensorflow
[10]https://www.yuque.com/easytransfer/cn/geiy58#IKdVp

the specific-shared (or shared-private) architecture. Furthermore, DRSS can be coupled with adversarial learning [21] to further boost the model performance.

Instance-based TL. Due to the domain difference, a vanilla TL method may suffer from the negative transfer. Instance-based TL methods seek to mitigate negative transfer by re-weighting source samples so that data from the source domain and the target domain would share a similar data distribution [4, 12, 35]. The TL module is typically considered as a sub-module of the data selection framework [35]. Therefore, the TL module needs to be retrained repetitively to provide sufficient updates to the data selection framework which may suffer from a long training time when applied to neural TL models.

EasyTransfer supports two in-house developed instance-based TL algorithms, namely Minimax Game-based TL (MGTL) [45] and Reinforced TL (RTL) [30]. RTL is the first work to leverage reinforcement learning to select high-quality source data to help the TL process. The reinforced data selector serves as an agent to interact with the environment created by the TL module. The selector "acts" on source data instances to select a subset of source data, and feeds them together with target data to the TL environment. The performance gap serves as a delayed reward to update the RL policy. MGTL addresses the sparse reward issue in RTL by providing an additional domain discriminator to help the TL module learn domain-invariant features and also providing immediate rewards to guild the RL policy. In the experiments, we find both RTL and MGTL are shown to more efficient than the basic instance-based deep TL method such as [35].

Model-based TL. Model-based TL, especially learning a light student model using knowledge distillation, is an important aspect of TL for real-time deployment. EasyTransfer is equipped with many knowledge distillation methods [3, 9, 13, 36, 41] to compress a big model (e.g. 12-layer BERT) to a small model (e.g. 2-layer BERT or CNN). Furthermore, we develop the task-adaptive BERT compression algorithm named AdaBERT [3] with differentiable neural architecture search, which achieves significant inference speedup and model size reduction.

One disadvantage of previous studies is that they mostly focus on single-domain only, which ignores the transferable knowledge from other domains. We develop a meta-knowledge Distillation (MetaKD) framework [26] to build a meta-teacher model that captures transferable knowledge across domains and passes such knowledge to students. Experiments on public multi-domain NLP tasks show the effectiveness and superiority of the proposed MetaKD framework.

Meta Learning. Apart from the above deep TL algorithms, EasyTransfer is equipped with the ability of meta learning to improve the performance of domain-level and task-level knowledge transfer for large-scale PLMs. For example, the MetaFT algorithm [46] is proposed to learn a "meta-learner" based on PLMs, aiming to solve a group of NLP tasks across different domains.

Additionally, we further consider the situation where there exist several similar NLP tasks related to distant domains with different class label sets. The Meta Distant Transfer Learning (MetaDTL) algorithm is proposed and integrated into EasyTransfer to learn the cross-task "meta-knowledge" by modeling the implicit relations among multiple tasks and classes, and then selectively learning the task-agnostic cross-task meta-knowledge. After the "meta-learner" is obtained, the model can be quickly adapted to a specific task with better performance by model fine-tuning. Experiments on public datasets show that MetaFT and MetaDTL are capable of learning cross-domain and cross-task knowledge for downstream tasks.

Table 5: The split of multiple domains in MNLI and Amazon Review datasets for evaluating MetaKD and MetaFT.

Dataset	Domain	#Train	#Dev	#Test
MNLI	Fiction	69,613	7,735	1,973
	Government	69,615	7,735	1,945
	Slate	69,575	7,731	1,955
	Telephone	75,013	8,335	1,966
	Travel	69,615	7,735	1,976
Amazon Review	Book	1,631	170	199
	DVD	1,621	194	185
	Electronics	1,615	172	213
	Kitchen	1,613	184	203

5 EXPERIMENTS

In this section, we empirically examine the effectiveness and efficiency of the EasyTransfer toolkit in both open datasets and industrial-scale applications.

5.1 Tasks and Datasets

In the experiments, we have examined the EasyTransfer framework over the following tasks and datasets:

- **Paraphrase Identification.** This is a task to examine the relationship, i.e., a paraphrase or not, between two input text sequences. We treat the Quora question pairs [11] as the source domain and a paraphrase dataset made available in CIKM AnalytiCup 2018 [12] as the target. The former is a large-scale dataset that covers a variety of topics, while the latter consists of question pairs in the E-commerce domain. We follow the study in [30] for data pre-processing.
- **Sentiment Analysis.** We use a popular sentiment analysis dataset Amazon Reviews [2], which is widely used in multi-domain text classification tasks. The reviews are annotated as positive or negative. For each domain, there are 2,000 labeled reviews. We randomly split the data into training, development, and testing sets.
- **Text Matching.** For text matching, we use two open text matching datasets to evaluate our method: MNLI [48] and SciTail [14]. MNLI is a large crowd-sourced corpora for textual entailment recognition with diverse text sources. SciTail is a recently released textual entailment dataset in the science domain. The SciTail dataset is smaller than MNLI but with more diversity in terms of linguistic variations. Note that each sample in MNLI is a sentence pair that contains a premise, a hypothesis, and a relation label of ENTAILMENT,

[11]https://www.kaggle.com/c/quora-question-pairs
[12]https://tianchi.aliyun.com/competition/introduction.htm?raceId=231661

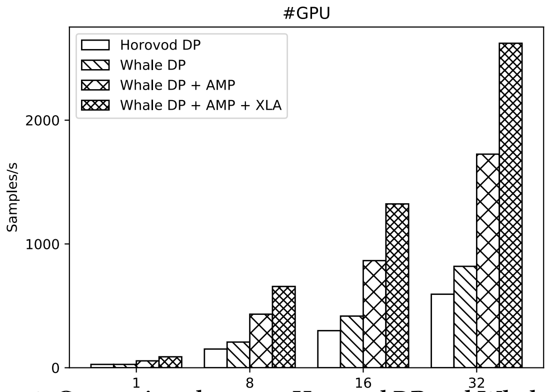

Figure 2: Comparison between Horovod DP and Whale DP (w. or w/o AMP and XLA).

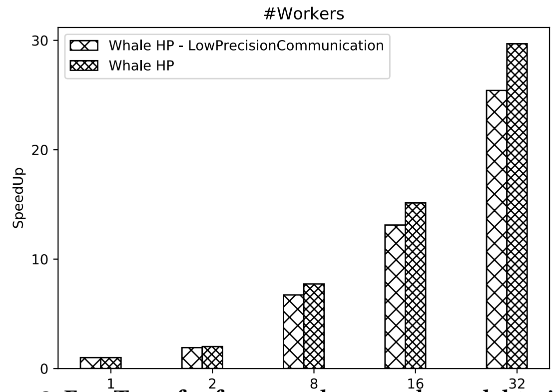

Figure 3: EasyTransfer for super large-scale model training (10 billion parameters).

NEUTRAL, or CONTRADICTION. However, the labels in SciTail only consist of ENTAILMENT and NEUTRAL.

- **Item Recommendation.** We have collected one-week user click-through data from two E-commerce websites, namely Taobao and Qintao. These two platforms share similar users and item features, but data distributions are different. As for the training data, Taobao has 52000 million samples and Qintao has 350 million samples.

The data statistics of MNLI and Amazon Reviews for evaluating MetaKD and MetaFT are shown in Table 5.

5.2 Distributed Training Performance

5.2.1 Efficient Data Parallelism. We use the ALBERT xxLarge model to conduct distributed training using Data Parallelism (DP). We compare our default distributed training engine Whale with an efficient distributed deep learning training framework Horovod [13] in Figure 2. Clearly, the Whale DP has better performance than Horovod DP, which shows the effectiveness of the proposed communication optimization as described in Section 3.2. Furthermore, EasyTransfer supports auto mixed precision (AMP) and Accelerated Linear Algebra (XLA) which can help to further improve the total throughput (from around 1000 to 2500). As a result, our final DP method achieves a 4.4 times speedup compared to the community version of distributed training.

5.2.2 Efficient Hybrid Parallelism for Super Large Models. When it comes to super large model pre-training, memory pressure becomes the main issue. For example, the T5-11B model, which consists of 24 layer encoder and decoder, intermediate size 65536 with 128-headed attention producing a model with about 11 billion parameters that can't be fit into one single GPU. Therefore, we use Whale HP to split the model into 8 GPUs within a single worker and train it with multiple workers. As shown in Figure 3, if we don't use low-precision weights (FP16) for communication, we observe a good speedup w.r.t. the number of workers. With low-precision communication, our method achieves a near-liner speedup, e.g. around 30 times speedup with 32 workers. This shows our framework is with a highly efficient training performance for super-large models. Last, Whale also supports Mixture-of-Experts based model sharding [38], which can train models with up to 100 billion parameters.

[13] https://github.com/horovod/horovod

Table 6: Performance of PLMs on the SuperGLUE benchmark. Bold texts denote best results in each column. "g-" means the pre-trained weights are from "google-bert".

Model	SuperGLUE Datasets					Avg
	CB	COPA	BoolQ	WiC	WSC	
g-bert-base	75.0	61.0	74.5	69.1	63.5	68.6
g-bert-large	73.2	62.0	79.1	69.8	**65.4**	69.9
g-albert-large	85.7	68.0	79.2	72.7	63.5	73.8
g-albert-xxlarge	83.9	**85.0**	84.5	**75.2**	63.5	78.4
pai-albert-xxlarge	**85.7**	84.0	**85.4**	74.6	63.5	**78.6**

5.3 PLM Performance

Benchmark. In order to evaluate the reliability of our PLMs in EasyTranfer, we evaluate five downstream tasks from the SuperGLUE benchmark. For all tasks, we use a limited hyper-parameter search, with batch sizes in {8, 16, 32} and learning rate in {$1e-5$, $3e-5$, $5e-5$}. We perform early stopping based on each task's evaluation metric on the development set. In this setting, we report the median development set results for each task over three random initialization. Table 6 shows the performance of baseline models on SuperGLUE benchmarks. As we can see, pai-albert-xxlarge can get comparable or better performance than others. This shows the EasyTransfer framework is reliable and can pre-train models with similar performance compared to the open-source framework.

RSA-based Transformers. We further examine the effectiveness of our proposed RSA-based method on GLUE tasks in Table 7. We find that we can save approximately 25% of the original training time, from 143 hours to 105 hours for pre-training BERT and from 180 to 135 for pre-training T5. It is due to the fact that RSA with a sparse ratio 1/4 (k=4) saves memory dramatically. Hence it takes less training time to converge to a reasonable loss. Both RSA-BERT and RSA-T5 have a similar downstream performance to that of the original BERT and T5 models, respectively. It shows that the RSA approach is generic to the transformer-based architecture. In

Table 7: Performance of PLMs on the CLUE benchmark. Here, "hour" means the time for model pre-training.

Model	GLUE Datasets				Avg	Hour
	SST-2	QQP	MRPC	RTE		
BERT	92.1	90.6	85.8	67.2	83.9	143
RSA-BERT$_{k2}$	92.2	90.3	86.5	65.7	83.7	137 (-4%)
RSA-BERT$_{k4}$	91.7	90.1	84.8	66.1	83.2	**105** (-27%)
T5	92.6	90.6	86.0	67.5	84.2	180
RSA-T5$_{k2}$	92.1	90.5	86.1	66.7	83.9	170 (-6%)
RSA-T5$_{k4}$	90.3	89.4	86.0	66.0	82.9	**135** (-25%)

a nutshell, the RSA method reduces the memory footprint and computation FLOPs and also keeps the model quality comparable with the original transformer-based models.

Online Applications. The RSA-based transformers have been widely used in real-world applications inside Alibaba. Most notably, RSA-BERT has been applied on query-item retrieval in Alibaba.com. The key task is to match queries and item titles in the e-commerce search engine of Alibaba.com. As a result, RSA-BERT outperforms the online production method (a DNN based query-item matching method) and has improved around 3% in terms of AUC.

5.4 Evaluations of Feature-based TL

We evaluate the performance of feature-based TL algorithms on Paraphrase Identification (PI) [50] in Table 8. We can observe that for all the five target domains, Src-Only performs much worse than Tgt-Only. The average performance of Mixed is even worse than Tgt-Only. This implies that the source domain is quite different from all the target domains. Simply mixing the training data in two domains may lead to the model overfitting to the large domain. In addition, it is observed that the widely used model fine-tuning method performs slightly better than Tgt-Only in most cases, which shows that pre-training the model on the source domain is beneficial. Moreover, in all the five domains, the performance of two existing transfer learning frameworks FS and SS are both 1.9% better than that of Tgt-Only, which proves their usefulness. Furthermore, our featured methods DRSS and DRSS-Adv can achieve better results than other methods, which shows the importance of domain-relationship learning in feature-based TL.

Online Applications. We deploy DRSS-Adv online in the AliMe assistant chatbot [18]. The original question-answering module in AliMe is a GBDT model with a set of semantic matching features including TF-IDF scores, word2vec scores, TextCNN scores, etc. We propose to train GBDT by treating the prediction score of our DRSS model as another feature, with the resulting method as GBDT+DRSS. For online evaluation, we randomly sample 2750 questions, where 1317 questions are answered by GBDT and 1433 questions are answered by GBDT-DRSS. We asked business analysts to annotate if the nearest question returned by models expresses the same meaning as the query question. The Precision@1 of GBDT-DRSS is 18.8% higher than that of GBDT (53.9% vs. 68.1%).

Table 8: Results of feature-based TL methods on Paraphrase Identification task.

Method	Prec@1	Rec@1	F1@1	ACC	AUC
Tgt-Only	71.7	55.1	62.3	79.2	82.2
Src-Only	61.9	36.8	46.1	71.9	68.6
Fine-Tune	71.3	56.7	63.2	79.0	82.5
FS [25]	73.4	59.5	65.7	79.7	83.1
SS [21]	74.4	60.1	66.5	80.0	83.7
SS-Adv [21]	74.3	60.3	66.6	80.8	84.2
DRSS	**75.7**	60.8	67.4	**81.2**	84.7
DRSS-Adv	75.3	**62.0**	**68.0**	80.9	**84.9**

Table 9: Evaluation of instance-based TL methods.

Method	Item Recommendation		Text Matching	
	ACC	AUC	ACC	AUC
Src-only	89.1	69.1	71.1	70.9
Tgt-only	90.1	70.3	73.0	76.6
TL Method [24, 25]	91.0	72.1	74.5	80.4
Ruder and Plank [35]	91.2	73.3	75.2	80.6
RTL	91.3	73.3	76.7	81.6
MGTL	**91.5**	**74.5**	**77.8**	**82.5**

5.5 Evaluations of Instance-based TL

We evaluate the two in-house intance-based TL algorithms in EasyTransfer, namely MGTL [45] and RTL [30]. We also compare these methods with a source-data only method, a target-data only method, representative TL algorithms [24, 25] and an instance selection method with Bayesian optimization [35] on the task of item recommendation and text matching as studied in [30] in terms of ACC and AUC. As shown in Table 9, a few important observations are as follows. First, Src-only performs worse than Tgt-only, which means source data is close to but different from target data, i.e. there is a domain shift between the source and target domain. In this case, simply using data from the source domain is not satisfactory. Second, the TL method improves the Tgt-only model which means the TL method can help to leverage information from the source domain to help the target domain. Furthermore, the instance selection method in [35] can further improve the TL model. MGTL achieves better performance than the method in [35] and also the vanilla RTL method, which further shows our immediate rewards and delayed rewards are helpful for the tasks. In all, the comparison with other competing methods shows the advantage of the reinforced selector based TL algorithms.

Online Applications. We have deployed MGTL on the platform Qintao[14] to leverage information from Taobao[15] for the task of item search, as the platform Taobao has 100x more data instances than Qintao. We compare MGTL with a baseline model which is a degenerated version of our model that does not consider source

[14]https://qintao.taobao.com
[15]https://taobao.com

Table 10: Evaluation of MetaKD over Amazon reviews (with four domains) in terms of accuracy (%).

Method	Books	DVD	Elec.	Kit.	Avg.
BERT-s	87.9	83.8	89.2	90.6	87.9
BERT-mix	89.9	85.9	90.1	**92.1**	89.5
BERT-mtl	90.5	86.5	91.1	91.1	89.8
Meta-teacher	92.5	87.0	91.1	89.2	89.9
BERT-s → TinyBERT	83.4	83.2	89.2	91.1	86.7
BERT-mix → TinyBERT	88.4	81.6	89.7	89.7	87.3
BERT-mtl → TinyBERT	90.5	81.6	88.7	90.1	87.7
MetaKD	**91.5**	**86.5**	**90.1**	89.7	**89.4**

instance selection. We have run 7-day A/B testing on the click-through-rate of the models. Our method outperforms the online model with 3%+ improvement on average. This brings to an increase of 1.5%+ of the total gross merchandise volume.

5.6 Evaluations of Model-based TL

We further proceed to evaluate MetaKD [26] on Amazon Reviews [2]. Table 10 shows the general testing performance of baselines and MetaKD, in terms of accuracy. Here, BERT-s refers to a single BERT teacher trained on each domain, BERT-mix is one BERT teacher trained on the mix of all domain data and BERT-mtl is one teacher trained by multi-task learning over all domains. For distillation baselines, "→ TinyBERT" means using the KD method described in the TinyBERT paper [13] to distill the corresponding teacher model. For each method, the teacher model is a BERT-base (Total Parameters=110M) model and the student model is a BERT-Tiny (Total Parameters=14.5M) model. Compared to all the baseline teacher models, MetaKD achieves the highest accuracy. Our method significantly reduces the model size while preserving a similar performance. Specifically, we reduce the model size to 7.5x smaller (BERT to TinyBERT) with only a minor performance drop (from 89.9% to 89.4%).

Online Applications. A typical way of building an efficient and effective model for real-world application is to first fine-tune a large model to achieve good performance first, and then distill it to a smaller efficient model. Inside Alibaba, EasyTransfer has provided a widely-used model distillation service to many applications that require low latency, such as user intent detection in AliMe assistant chatbot, malicious reviews detection on e-commerce platforms, etc.

5.7 Evaluations of Meta Learning

The MetaFT algorithm leverages the power of the meta-learner, hence is highly effective in the few-shot learning setting. Take the multi-domain MNLI dataset [48] for an example. For each of the five domains, we only use 5%, 10%, and 20% of the original dataset for model fine-tuning. The prediction accuracy scores with and without the MetaFT algorithm are compared, with BERT-base as the underlying PLM. The results are in Table 11.

As MetaFT only considers learning meta-learners across similar domains, we consider a more challenging case to evaluate MetaDTL. In this experiment, we randomly sample only 1%, 2%, 5%, 10%, and 20% of the original MNLI training data to fine-tune the BERT model. The SciTail [14] training set is used for knowledge transfer. We list the results on the MNLI testing set with and without MetaDTL training in Table 12. The results produced by Meta-FT and single-task fine-tuning are also compared. As seen, MetaDTL improves the performance regardless of the percentages of the training sets. It has a larger increase in accuracy on smaller training sets (4.0% increase on 1% of the training set vs. 1.0% increase on 20%). In summary, MeteFT and MetaDTL are capable of training meta-learners across domains and tasks, suitable for tasks with very little training data.

Table 11: Evaluation of MetaFT on natural language inference over few-shot MNLI in terms of accuracy (%).

Domain	MetaFT?		MetaFT?		MetaFT?	
	No	Yes	No	Yes	No	Yes
Ratio of training set	5%		10%		20%	
Telephone	70.5	74.7	74.1	76.4	75.9	79.8
Government	76.5	78.1	78.8	81.0	80.5	82.9
Slate	64.2	69.8	67.6	71.8	71.8	74.1
Travel	71.9	75.4	74.8	78.1	78.3	80.3
Fiction	69.7	73.8	73.3	76.6	76.2	78.4
Average	70.5	**74.4**	73.7	**76.8**	76.5	**79.1**

Table 12: Comparison between MetaFT and MetaDTL on the testing set when only part of the MNLI training set is employed in terms of accuracy (%).

Percentage	Single-task	MetaFT	MetaDTL
1%	62.5	64.1	**66.5 (+4.0%)**
2%	67.5	68.2	**69.8 (+2.3%)**
5%	72.8	73.8	**74.2 (+1.4%)**
10%	75.8	76.2	**77.6 (+1.8%)**
20%	80.4	80.8	**81.4 (+1.0%)**

Online Applications. The "meta-learners" trained by EasyTransfer are frequently applied in various applications in Alibaba, especially for emerging domains with few training data. For example, in the AliMe assistant chatbot, there are over one thousand similar user intent classification tasks for different domains and businesses. By applying the "meta-learner" fine-tuned on PLMs, the overall accuracy is improved by over 8%, compared to the online system.

6 CONCLUSION AND FUTURE WORK

In this paper, we introduced EasyTransfer, a toolkit that is designed to make it easy to develop deep TL algorithms for NLP applications. To cope with the development of large pretrained models, EasyTransfer is built with a scalable architecture to support large model training and inference. And to meet the need of diverse real-world applications, EasyTrasfer supports a rich family of TL algorithms, and has been used to support many (20+) business scenarios in Alibaba. It has been integrated into Alibaba Cloud to support many external business needs. The toolkit has been open-sourced to promote research for deep TL and NLP applications.

REFERENCES

[1] Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer. *arXiv:2004.05150* (2020).

[2] John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In *ACL*.

[3] D. Chen, Y. Li, M. Qiu, Z. Wang, B. Li, B. Ding, H. Deng, J. Huang, W. Lin, and J. Zhou. 2020. AdaBERT: Task-Adaptive BERT Compression with Differentiable Neural Architecture Search. In *IJCAI*. 2463–2469.

[4] Minmin Chen, Kilian Q Weinberger, and John Blitzer. 2011. Co-training for domain adaptation. In *NeurIPS*. 2456–2464.

[5] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating Long Sequences with Sparse Transformers. *arXiv:1904.10509* (2019).

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

[7] Dehong Gao, Linbo Jin, Ben Chen, Minghui Qiu, Peng Li, Yi Wei, Yi Hu, and Hao Wang. 2020. FashionBERT: Text and Image Matching with Adaptive Loss for Cross-modal Retrieval. In *SIGIR*.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. 770–778.

[9] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*.

[10] Timothy M. Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J. Storkey. 2020. Meta-Learning in Neural Networks: A Survey. *arXiv:2004.05439* (2020).

[11] Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *ACL*. 328–339.

[12] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex J Smola. 2007. Correcting sample selection bias by unlabeled data. In *NeurIPS*. 601–608.

[13] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for Natural Language Understanding. In *EMNLP (Findings)*. 4163–4174.

[14] Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. SciTaiL: A Textual Entailment Dataset from Science Question Answering. In *AAAI*. 5189–5197.

[15] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR*.

[16] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. *arXiv:1810.00825* (2019).

[17] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *arXiv:1910.13461* (2019).

[18] F. Li, M. Qiu, H. Chen, X. Wang, X. Gao, J. Huang, J. Ren, Z. Zhao, W. Zhao, L. Wang, G. Jin, and W. Chu. 2017. AliMe Assist : An Intelligent Assistant for Creating an Innovative E-commerce Experience. In *CIKM*.

[19] Gen Li, Nan Duan, Yuejian Fang, Ming Gong, Daxin Jiang, and Ming Zhou. 2019. Unicoder-VL: A Universal Encoder for Vision and Language by Cross-modal Pre-training. *arXiv:1908.06066* (2019).

[20] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. 2017. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887* (2017).

[21] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial Multi-task Learning for Text Classification. In *ACL*. 1–10.

[22] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692* (2019).

[23] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks. *arXiv:1908.02265* (2019).

[24] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural Language Inference by Tree-Based Convolution and Heuristic Matching. In *ACL*.

[25] Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How Transferable are Neural Networks in NLP Applications?. In *EMNLP*.

[26] Haojie Pan, Chengyu Wang, Minghui Qiu, Yichang Zhang, Yaliang Li, and Jun Huang. 2021. Meta-KD: A Meta Knowledge Distillation Framework for Language Model Compression across Domains. In *ACL/IJCNLP*. 3026–3036.

[27] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* 22, 10 (2010), 1345–1359.

[28] Huimin Peng. 2020. A Comprehensive Overview and Survey of Recent Advances in Meta-Learning. *arXiv:2004.11149* (2020).

[29] Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen tau Yih, Sinong Wang, and Jie Tang. 2020. Blockwise Self-Attention for Long Document Understanding. *arXiv:1911.02972* (2020).

[30] Chen Qu, Feng Ji, Minghui Qiu, Liu Yang, Zhiyu Min, Haiqing Chen, Jun Huang, and W Bruce Croft. 2019. Learning to selectively transfer: Reinforced transfer learning for deep text matching. In *WSDM*. 699–707.

[31] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *arXiv* (2018).

[32] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[33] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67.

[34] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. FitNets: Hints for Thin Deep Nets. In *ICLR*.

[35] Sebastian Ruder and Barbara Plank. 2017. Learning to select data for transfer learning with Bayesian Optimization. In *EMNLP*. 372–382.

[36] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv:1910.01108* (2019).

[37] Tobias Schnabel and Hinrich Schütze. 2014. FLORS: Fast and Simple Domain Adaptation for Part-of-Speech Tagging. *Trans. Assoc. Comput. Linguistics* 2 (2014), 15–26.

[38] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. arXiv:1701.06538 [cs.LG]

[39] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. 2018. Wasserstein Distance Guided Representation Learning for Domain Adaptation. In *AAAI*. 4058–4065.

[40] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. 2020. VL-BERT: Pre-training of Generic Visual-Linguistic Representations. *arXiv:1908.08530* (2020).

[41] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient Knowledge Distillation for BERT Model Compression. In *EMNLP-IJCNLP*. 4322–4331.

[42] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. 2018. A Survey on Deep Transfer Learning. In *ICANN*. 270–279.

[43] Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling Task-Specific Knowledge from BERT into Simple Neural Networks. *arXiv: 1903.12136* (2019).

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*. 5998–6008.

[45] B. Wang, M. Qiu, X. Wang, Y. Li, Y. Gong, X. Zeng, J. Huang, B. Zheng, D. Cai, and J. Zhou. 2019. A Minimax Game for Instance based Selective Transfer Learning. In *SIGKDD*.

[46] C. Wang, M. Qiu, J. Huang, and X. He. 2020. Meta Fine-Tuning Neural Language Models for Multi-Domain Text Mining. In *EMNLP*.

[47] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-Attention with Linear Complexity. *arXiv:2006.04768* (2020).

[48] Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *NAACL*.

[49] Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2017. Transfer Learning for Sequence Tagging with Hierarchical Recurrent Networks. In *ICLR*.

[50] Jianfei Yu, Minghui Qiu, Jing Jiang, Jun Huang, Shuangyong Song, Wei Chu, and Haiqing Chen. 2018. Modelling Domain Relationships for Transfer Learning on Retrieval-based Question Answering Systems in E-commerce. In *WSDM*.

[51] Zhuo Zhang, Chao Li, Yangyu Tao, Renyu Yang, Hong Tang, and Jie Xu. 2014. Fuxi: a Fault-Tolerant Resource Management and Job Scheduling System at Internet Scale. *Proc. VLDB Endow.* 7, 13 (2014), 1393–1404.

[52] Zhen Zheng, Pengzhan Zhao, Guoping Long, Feiwen Zhu, Kai Zhu, Wenyi Zhao, Lansong Diao, Jun Yang, and Wei Lin. 2020. Fusionstitching: boosting memory intensive computations for deep learning workloads. *arXiv:2009.10924* (2020).

[53] Fuzhen Zhuang, Lang Huang, Jia He, Jixin Ma, and Qing He. 2017. Transfer Learning with Manifold Regularized Convolutional Neural Network. In *KSEM*.

[54] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. 2019. A Comprehensive Survey on Transfer Learning. *arXiv:1911.02685* (2019).