Preventing Catastrophic Forgetting in Continual Learning of New Natural Language Tasks

Sudipta Kar*
Amazon
Seattle, WA, USA
sudipkar@amazon.com

Giuseppe Castellucci
Amazon
Seattle, WA, USA
giusecas@amazon.com

Simone Filice
Amazon
Tel Aviv, Israel
filicesf@amazon.com

Shervin Malmasi
Amazon
Seattle, WA, USA
malmasi@amazon.com

Oleg Rokhlenko
Amazon
Seattle, WA, USA
olegro@amazon.com

ABSTRACT

Multi-Task Learning (MTL) is widely-accepted in Natural Language Processing as a standard technique for learning multiple related tasks in one model. Training an MTL model requires having the training data for all tasks available at the same time. As systems usually evolve over time, (e.g., to support new functionalities), adding a new task to an existing MTL model usually requires retraining the model from scratch on all the tasks and this can be time-consuming and computationally expensive. Moreover, in some scenarios, the data used to train the original training may be no longer available, for example, due to storage or privacy concerns.

In this paper, we approach the problem of incrementally expanding MTL models' capability to solve new tasks over time by *distilling* the knowledge of an already trained model on n tasks into a new one for solving $n + 1$ tasks. To avoid catastrophic forgetting, we propose to exploit unlabeled data from the same distributions of the old tasks. Our experiments on publicly available benchmarks show that such a technique dramatically benefits the distillation by preserving the already acquired knowledge (i.e., preventing up to 20% performance drops on old tasks) while obtaining good performance on the incrementally added tasks. Further, we also show that our approach is beneficial in practical settings by using data from a leading voice assistant.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**; **Online learning settings**; **Semi-supervised learning settings**.

KEYWORDS

continual learning, catastrophic forgetting, text classification

ACM Reference Format:
Sudipta Kar, Giuseppe Castellucci, Simone Filice, Shervin Malmasi, and Oleg Rokhlenko. 2022. Preventing Catastrophic Forgetting in Continual Learning of New Natural Language Tasks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3534678.3539169

*Work done during internship.

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08.
https://doi.org/10.1145/3534678.3539169

1 INTRODUCTION

In recent years, voice assistants, like Alexa or Siri, have become very popular. They exploit Natural Language Processing (NLP) capabilities (e.g., intent classification, slot filling, etc.) to provide useful functionalities to the users, e.g., asking questions, getting the latest news, getting weather updates, etc. Multi-Task Learning (MTL) models are often preferred in such systems as they have multiple advantages: (*i*) MTL normally leads to a better generalization of the model by exploiting the domain-specific information in the training signals of the related tasks [27]; (*ii*) MTL models are easier to deploy and maintain when compared to multiple single task models; (*iii*) MTL models can reduce overall inference latency as they solve multiple tasks in a single inference step.

Modern systems are continually updated over time to support new functionalities, e.g., new intents and slots, or even new tasks. In MTL, accommodating new tasks usually means training a new model from scratch by using all the past and new training data for all the tasks to be supported. This process can be both time-consuming and computationally expensive. Moreover, this may not always be possible in practical settings as the past data might no longer be available. There may be storage, regulatory, customer, business, or privacy-related constraints, as well as issues of missing and corrupted data. In such cases, the original training data cannot be used, making a full re-training of the MTL model infeasible.

Continual Learning (CL) proposes a viable solution to enable models to keep on learning over time [4]. In CL, a model is updated by only considering the newest task's training data. Thus, a major concern in CL is that it can cause the so-called Catastrophic Forgetting (CF) [8, 22], where performance drops in the previously learned tasks due to the different data distribution of the new training set [17]. A solution to this problem is to use Knowledge Distillation (KD) [10], to "distill" (i.e., transfer[1]) the knowledge from a teacher model to a student model. The student learns a task from the soft targets (i.e., output scores) produced by the teacher. In this way, the new model can exploit the teacher's uncertainty contained in

[1]Transfer learning usually refers to using a model trained on a source domain to help target domains learning, but it is not continual and it has no knowledge retention mechanism.

its scores as well. KD is often applied for model compression to shrink a large model into a smaller but similarly performing one. In Computer Vision, KD has been successfully adopted in a CL setting (e.g., Li and Hoiem [19]) where a model acts as the teacher providing soft targets for n tasks to a student model that is trained to solve the same set of tasks plus a new one. Applying such a technique in NLP for continual learning is not straightforward, as the data distribution for the different tasks can vary greatly. For example, let us assume we have an intent classification model trained to support an initial set of intents. If we need to extend the model to support a very different set of intents, we might expect a drastic change in the incoming utterance distribution. This can negatively impact the application of the standard KD technique.

We propose a semi-supervised continual learning solution for extending a model already trained on n tasks to solve also a new one, by using only the new tasks' training data as annotated material. In order to prevent the catastrophic forgetting phenomenon, we exploit a set of *unlabeled* material from the data distributions of the existing tasks. In particular, we aim to distill the old model's (the teacher) knowledge to a new one (the student) by using such unlabeled data, which is supposed to better resemble the old training material data distribution.

We experiment with our approach on a set of text classification tasks from the GLUE benchmark [33] using BERT-based models. We compare different continual learning strategies, ranging from simple re-training to traditional KD under the assumption that the training data of the previous tasks is no longer available. Experimental results show that our proposed approach can effectively learn sequences of tasks, where at each stage there is no, or very little, forgetting of the previous tasks (i.e., preventing up to 20% performance drops on old tasks) while still being capable of learning to solve a new task. Moreover, we perform similar experiments in a real setting by exploring a dataset collected from a real voice assistant. We show that our proposed approach allows updating an existing MTL model for the Intent Classification task on different domains while preventing catastrophic forgetting. Our contributions can be summarized as follows:

- We show that we can minimize the catastrophic forgetting phenomenon during incremental learning of NLP tasks.
- With our CL approach, we achieve comparable results with fully retrained MTL models.
- We show that the general knowledge of pre-trained language models is preserved even after several fine-tuning phases.
- Our approach works both on publicly available benchmarks (i.e., GLUE) and real scenarios over real voice assistant data.

The rest of the paper is organized as follows: Section 2 discusses the related works; Sections 3 and 4 present the continual learning approach and the experiments, respectively. Finally, Section 5 discusses our conclusions and future work.

2 RELATED WORK

Multi-Task Learning (MTL) [2] models achieve impressive results in several domains [27], including Natural Language Understanding [5, 21, 24]. However, MTL models require to be trained from scratch every time there is the need to adapt them to new tasks, which can be infeasible in many scenarios. In contrast, Continual Learning (CL) (also referred to as lifelong learning [4]) studies the problem of learning from a stream of data and extending the acquired knowledge [16] over time. The stream can change during time by an evolution in the input distribution or by incorporating new tasks. A key challenge in adding new tasks with CL strategies is to avoid the Catastrophic Forgetting (CF) [8] phenomenon.

CL has been explored mostly in Computer Vision [1, 9, 14, 19, 25, 26, 30, 32]. Few works involve continual learning in the NLP domain. In [28] the authors introduce the PGN architecture where new copies of a model are issued for each new task. Elastic Weight Consolidation [15] is another CL technique, where task-specific constraints on the model weights are added to prevent the catastrophic forgetting phenomenon. In the slot filling task, the work of Shen et al. [29] proposes a strategy to progressively increase the slot types the model can recognize. In [20], the authors train sentence encoders with unsupervised methods to continually learn features from new corpora. Similarly, Jin et al. [13] discuss the problem of continually adapting large pre-trained language models to new settings. In [18], the authors address the open and growing vocabulary problem in a sequence-to-sequence framework. Huang et al. [11] propose a regularization approach based on information disentanglement by separating the hidden space in task general and task-specific sub-spaces by utilizing auxiliary tasks.

Most of these works are either not directly applicable to the setting proposed here or are unfeasible with large pre-trained models. In [23] a CL approach using a KD mechanism is proposed for Named Entity Recognition problems, where the authors incrementally train a sequence tagging model to support new entities over time. Castellucci et al. [3] used distillation to continually adapt to new languages. In this work, instead, we aim to adopt an already trained model to new NLP tasks with large pre-trained models like BERT by exploiting KD in a teacher-student setting [30], where we make use of unlabeled text data to prevent forgetting of the previously learned tasks.

3 CONTINUAL LEARNING FOR NATURAL LANGUAGE TASKS

To continually learn new tasks for NLP, we exploit the KD [10] framework. Without loss of generality, let us assume that we have already trained a model θ_A to solve task A and we want to update it to learn how to also solve a new task B. As illustrated in Figure 1, we start by creating a copy of θ_A, by adding a new output layer for task B, i.e., θ_{AB}. The original θ_A and θ_{AB} models act as the teacher and the student in the KD framework, respectively. During training, we keep we θ_A frozen and we only update θ_{AB} with the objective of (*i*) learning the new task B from the training data and (*ii*) preserving the older task's knowledge by minimizing the loss function

$$\mathcal{L}_{AB} = \mathcal{L}_{KD_A} + \mathcal{L}_B.$$

Let us consider a set of training examples $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}$ is an input representation and $\boldsymbol{y}$ is a target category. The distillation loss is defined as $\mathcal{L}_{KD_A} = CE(\theta_A(\boldsymbol{x}), \theta_{AB}(\boldsymbol{x}))$, i.e., the cross-entropy (CE) between the class probability distribution of the student on task A and the soft targets derived from the teacher θ_A by applying a temperature-controlled softmax. The temperature-controlled

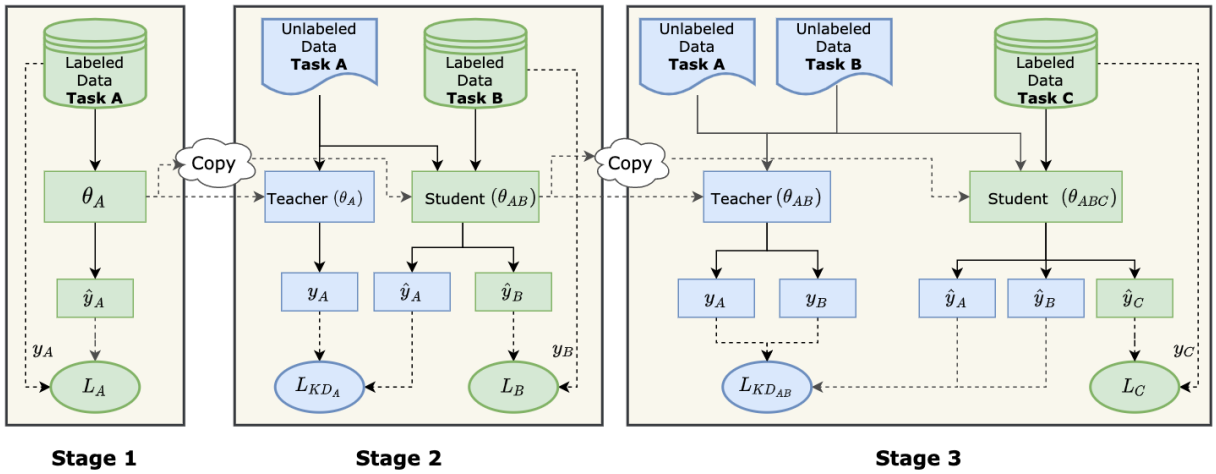

Figure 1: An illustration of incrementally learning three tasks through our proposed CL approach. Stage 1: A model θ_A is trained on `Task A` using a set of labeled data and by minimizing the Cross Entropy (CE) loss $\mathcal{L}_A$. Here y_A indicates the ground truth labels for `Task A` from the labeled dataset, and $\hat{y}_A$ indicates the predictions for `Task A` by θ_A. Stage 2: The trained θ_A model from Stage 1 is treated as a frozen `Teacher` model. A trainable `Student` copy θ_{AB} is created by adding a new classification head for `Task B`. The `Student` model is trained on the CE based task loss $\mathcal{L}_B$ for `Task B` and distillation loss $\mathcal{L}_{KD_A}$ for `Task A`. To compute $\mathcal{L}_{KD_A}$, a set of unlabeled data is used: `Teacher` model's predictions on such dataset for `Task A` are treated as soft labels and are used against the `Student` model's predictions. In this way, the `Student` learns to perform `Task B` and at the same time tries to keep `Task A`'s knowledge by distilling it from the `Teacher` model. Stage 3: The `Student` from stage 2 θ_{AB} acts as the frozen `Teacher`, and a `Student` copy θ_{ABC} is created to add `Task C`. The rest of the training process is similar to stage 2.

softmax converts the logit z_i of each class into a probability

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

where T controls the smoothness of the output distribution [10]. Using higher values of T will make the output distribution softer, and can be used to control the influence of the soft targets on the loss. Simultaneously, the model is minimizing also the new task loss, which is defined as $\mathcal{L}_B = CE(\boldsymbol{y}, \theta_{AB}(\boldsymbol{x}))$, i.e., the usual CE with respect to the annotated targets for task B. While $\mathcal{L}_B$ serves to let the student learn how to solve a new task, $\mathcal{L}_{KD_A}$ helps it in preventing catastrophic forgetting of the old one. In the standard application of KD to CL, $\mathcal{L}_{KD_A}$ is computed on the new task data: this assumes that the old and new tasks have the same data distribution. For instance, in Computer Vision, when models are trained to recognize an increasing number of object classes, the input images come from the same underlying distribution [30].

However, in NLP, tasks are typically defined on very different data distributions, and preventing the catastrophic forgetting when using only the new task data can be challenging. As an example, let us assume we want to add the answer selection task to a model trained for paraphrase identification. In this case, examples for the new task are (*question*, *candidate_answer*) pairs, which, intuitively, are not paraphrases. If we use this data to compute the distillation loss we never consolidate the model capability to recognize paraphrases but we bias it toward the non-paraphrase class.

3.1 Dealing with Different Data Distributions

We propose augmenting the KD learning process with a data distribution resembling the one used to train the teacher model to solve task A. Our assumption is that while the original training material for task A may no longer be available, we can still observe a stream of unlabeled data (U_A) from the same distribution. For example, the raw data on which the teacher model is applied for inference. Note that in case of storage constraints we can still process the data on the fly to accumulate gradients and run mini-batch training.

By doing so, the loss function $\mathcal{L}_{KD_A}$ represents the discrepancy between the teacher and student predictions for the old tasks on a set of unlabeled data. In practice, the unlabeled data U_A are automatically labeled by the teacher model θ_A to produce the soft targets dataset of task A. This dataset will be used to compute the loss $\mathcal{L}_{KD_A}$. Instead, a new labeled dataset for task B is used to compute $\mathcal{L}_B$. In this way, the student model should be able to minimize the discrepancy with the teacher on the old task (i.e., minimizing the catastrophic forgetting) while learning the new task.

This methodology can be easily extended to the general case where the teacher is already trained on n tasks and the student needs to solve a new task. In this setting, we need to prevent the catastrophic forgetting of n different tasks. We assume the availability of an unlabeled stream of data for each of the old tasks in order to compute the individual distillation losses. In this way, the student model will maintain the relevant knowledge to solve the n tasks by distilling it from the teacher on the unlabeled data stream, while also learning how to solve the new task on the labeled data. We will refer to this proposed approach of using unlabeled data for knowledge distillation as UKD throughout the rest of the paper.

Dataset: Multi-genre NLI Corpus (MNLI) **Train-Dev-Test Size:** 314k-79k-20k **Evaluation Metric:** Accuracy for *matched* and *mismatched* **Example:** ***S1****: i don't know um do you do a lot of camping* ***S2****: I know exactly.* ***Label:*** *Contradiction*
Dataset: Quora Question Pairs (QQP) **Train-Dev-Test Size:** 291k-73k-391k **Evaluation Metric:** F1/ Accuracy **Example:** ***Q1:*** *What are the best things to do in Hong Kong?* ***Q2:*** *What is the best thing in Hong Kong?* ***Label:*** *Duplicate*
Dataset: Question NLI (QNLI) **Train-Dev-Test Size:** 84k-21k-5.4k **Evaluation Metric:** Accuracy **Example:** ***Q:****What language did Tesla study while in school?* ***A:*** *Tesla was the fourth of five children.* ***Label:*** *Not entailment*
Dataset: Mircrosoft Research Paraphrase Corpus (MRPC) **Train-Dev-Test Size:** 2.9k-740-1.7k **Evaluation Metric:** F1/ Accuracy **Example:** ***S1:*** *Amrozi accused his brother, whom he called "the witness" , of deliberately distorting his evidence.* ***S2:*** *Referring to him as only "the witness", Amrozi accused his brother of deliberately distorting his evidence.* ***Label:*** *Paraphrase*
Dataset: Recognizing Textual Entailment (RTE) **Train-Dev-Test Size:** 2k-500-3k **Evaluation Metric:** Accuracy **Example:** ***S1:*** *No Weapons of Mass Destruction Found in Iraq Yet.* ***S2:*** *Weapons of Mass Destruction Found in Iraq.* ***Label:*** *Not Entailment*
Dataset: Stanford Sentiment Treebank (SST-2) **Train-Dev-Test Size:** 53.6k-13.4k-1.8k **Evaluation Metric:** Accuracy **Example:** *gorgeous and deceptively minimalist* ***Label:*** *Positive*

Table 1: Details about the tasks and their dataset. We treat the official development set as our test set and use 20% of the training set as our development set.

4 EXPERIMENTS AND RESULTS

4.1 Dataset

We conduct our continual learning experiments on the following datasets, which are part of the GLUE benchmark [33]: Multi-genre Natural Language Inference (MNLI; Williams et al., 2018), Quora Question Pairs (QQP; Iyer et al., 2017), Microsoft Research Paraphrase Corpus (MRPC; Dolan and Brockett, 2005), Question Natural Language Inference (QNLI), Recognizing Textual Entailment (RTE), and Stanford Sentiment Treebank (SST-2; Socher et al., 2013). These represent a diverse set of NLP text classification tasks with different data distributions, i.e., we can expect to observe the catastrophic forgetting phenomenon when updating an already trained model. In Table 1 we provide an overview of the tasks and their dataset. Notice that these datasets are also characterized by different sizes for the training material. We expect that catastrophic forgetting could be worse for those tasks where the training set is smaller.

We use Accuracy and F1 score to measure performance on different datasets, following the literature. For MNLI, we report accuracy for both *matched* (genres are the same as the training set) and *mismatched* (genres different than the training data) sets of the data.

4.2 Experimental Setup

In our experiments, we use the pre-trained BERT base uncased model [6] of the Transformers [35] library. To fine-tune BERT, we add an output layer (a dense layer followed by Softmax) for each task on top of the *CLS* token. We set the batch size to 64, the learning rate to 5e-6, the dropout to 0.1, and the temperature T to 2. To choose these parameters we ran a set of preliminary experiments with a grid search strategy. We finally trained each model for 20 epochs with early stopping with patience=3. The results reported in the following sections are based on the official validation set of each task. We used 20% of the training data for validation.

In our experiments we consider the following CL settings as baselines to compare with our proposed approach using unlabeled data for knowledge distillation (UKD), as described in Section 3.1.

- **Single Task (ST):** BERT is fine-tuned for only one task.
- **Multi Task Learning (MTL):** We fine-tune BERT through multi-task learning for all the tasks' training material. Notice that this setup assumes the availability of all the training material at the same time. Thus, we can consider MTL as an upper bound to the performance.
- **Output Layer (OL):** We take the model θ_A trained on task A and add a new classification head for task B. Then, we fine-tune only the new task output layer and freeze the rest of the network of the student model θ_{AB}.
- **Entire Model (EM):** We fine-tune the entire model by considering only the new task's loss. That means all of the parameters of BERT and the old tasks' output layer get updated to learn the new task.
- **Elastic Weight Consolidation (EWC):** We impose weight constraints on the model in line with the elastic weight consolidation (EWC) framework,[2] as suggested in Kirkpatrick et al. [15] while adapting the model to a new task.
- **Traditional Knowledge Distillation (TKD):** We fine-tune the entire model using both the new task and the traditional knowledge distillation [10] losses. The distillation loss for the previous tasks is computed on the new task's training data.

[2]We adopted the implementation available in https://github.com/GT-RIPL/Continual-Learning-Benchmark

In the experiments reported in section 4.3 and section 4.4, at each training step we use the same unlabeled data (our validation set) for distillation. While this violates our assumption of not being able to store data, it allows us to compare with published results using the development set of the GLUE benchmark. We randomly selected 20% of the training data to use as our validation set.

In section 4.5 we show a more realistic case where we use a different unlabeled set at each training stage, i.e., simulating the scenario where we are not allowed to store any data. In this case, results are not comparable with literature since we consider as labeled material only a subset of the training material at each training stage. Finally, in section 4.6 we report a set of experimental results obtained by applying our methodology on a dataset extracted from a real voice assistant's utterances.

4.3 Addition of a Second Task

We investigate different techniques for adding a new task to a model already trained on one task. We experimented with the three largest datasets in the GLUE benchmark: MNLI, QNLI, and QQP. We first fine-tune a pre-trained language model on one task. Then, we add another task to the model through the proposed technique without using any training labels for the first task. We present the results in Table 2, where we report also the performance of the baselines.

As hypothesized, when adding QNLI to a model fine-tuned on MNLI, in the OL setting the model is not able to learn the QNLI task by only adjusting the output layer weights (the new task accuracy is about 20 points below the single task setting (ST)). Alternatively, tuning the entire model (EM) causes catastrophic forgetting of the previous tasks, as demonstrated by the drop of about 10 points in accuracy in both the matched and mismatched settings. We note that the same pattern can also be observed for the other task pairs. Traditional Knowledge Distillation (TKD) brings a balance between these two methods as we observe good performances on both the first and second task. However, when the distillation loss is computed on the second task data (TKD row), the performance gap on the old tasks (w.r.t. ST or MTL) persists. For example, when adding QNLI to a model trained on MNLI data, the TKD approach is about 4 points less than the ST and the MTL systems. This is happening to various degrees to all the old tasks in all the pairs. Imposing constraints on the model weights, like in the EWC approach, is not effective. We argue that this can be caused by the usage of large and complex pre-trained language models. Given that the drop we observe for EWC is generally higher than other baselines, we will not report the EWC results in the following sections.

In sum, computing the distillation loss with our proposed UKD method largely mitigates the catastrophic forgetting issue and the capability of the model to learn the new task. When adding QNLI to an MNLI-trained model, the drop of the first task after at the second step is only about 0.2% when we use the MNLI unlabeled development set for distillation (TKD drop is about 3.5%). Additionally, QNLI accuracy when added as a new task is comparable with ST. This means that the model is retaining the general linguistic knowledge required to learn new tasks, while also preserving its knowledge on the old task. Moreover, it is worth noticing that UKD performances are comparable with MTL. We observe a similar trend in the reverse setting, where we add MNLI to a model fine-tuned on QNLI. Finally, this pattern is consistent in other task pairs as well (e.g., adding QQP to MNLI or QNLI).

Setting	**Avg.**	**MNLI** Acc.	**QNLI** Acc.	**QQP** F1 / Acc.
Single Task	87.5	83.9, 84.2	90.9	87.7 / 90.9
MTL [MNLI, QNLI]	86.0	83.7, 83.8	90.4	-
OL [MNLI, QNLI]	80.0	**83.9, 84.2**	71.8	-
EM [MNLI, QNLI]	79.2	73.2, 73.7	90.6	-
EWC [MNLI, QNLI]	73.8	65.6, 65.4	90.5	-
TKD [MNLI, QNLI]	84.0	80.3, 80.8	**90.8**	-
UKD [MNLI, QNLI]	**86.1**	83.5, 84.0	**90.8**	-
OL [QNLI, MNLI]	68.1	55.5, 58.0	90.9	-
EM [QNLI, MNLI]	73.2	**83.4, 84.1**	52.1	-
EWC [QNLI, MNLI]	73.5	83.7, 83.7	53.1	-
TKD [QNLI, MNLI]	85.5	83.1, 83.6	89.9	-
UKD [QNLI, MNLI]	**86.1**	83.5, 83.9	**91.0**	-
MTL [MNLI, QQP]	86.1	83.3, 83.2	-	87.3 / 90.4
OL [MNLI, QQP]	78.1	**83.9, 84.2**	-	67.3 / 76.8
EM [MNLI, QQP]	79.8	70.9, 71.9	-	86.6 / 90.0
EWC [MNLI, QQP]	80.0	70.8, 71.5	-	87.2 / 90.5
TKD [MNLI, QQP]	85.6	81.5, 82.0	-	**87.8 / 90.9**
UKD [MNLI, QQP]	**86.5**	83.5, 84.0	-	**87.7 / 90.9**
OL [QQP, MNLI]	70.6	51.4, 52.3	-	87.7 / 90.9
EM [QQP, MNLI]	81.1	83.0, 83.9	-	75.4 / 81.9
EWC [QQP, MNLI]	82.0	83.4, 83.8	-	77.3 / 83.3
TKD [QQP, MNLI]	85.1	82.5, 82.6	-	85.7 / 89.6
UKD [QQP, MNLI]	**86.0**	**83.1, 83.0**	-	87.2 / 90.6
MTL [QNLI, QQP]	89.1	-	90.6	86.7 / 90.0
OL [QNLI, QQP]	73.7	-	**90.9**	58.6 / 71.7
EM [QNLI, QQP]	78.7	-	59.3	86.8 / 90.1
EWC [QNLI, QQP]	79.5	-	61.0	87.1 / 90.4
TKD [QNLI, QQP]	89.3	-	89.7	**87.5 / 90.6**
UKD [QNLI, QQP]	**89.2**	-	90.1	87.0 / 90.4
OL [QQP, QNLI]	83.8	-	72.9	**87.7 / 90.9**
EM [QQP, QNLI]	84.4	-	89.5	79.0 / 84.8
EWC [QQP, QNLI]	83.2	-	89.8	75.8 / 84.0
TKD [QQP, QNLI]	88.2	-	**90.0**	85.3 / 89.4
UKD [QQP, QNLI]	**89.2**	-	89.9	87.2 / 90.5

Table 2: Comparison of different CL strategies to learn two tasks in different orders. Comparison of single task and multi-task (MTL) performance with different CL strategies. MT: Multi-task, OL: Fine-tune only the output layer, EM: fine-tune entire model, EWC: elastic weight consolidation [15], TKD: traditional knowledge distillation, UKD: knowledge distillation with unlabaled data. [MNLI, QNLI] means QNLI is added to an MNLI model. Best results are in bold. Two different accuracy scores are reported for the *matched* and *mismatched* sets of MNLI.

Setting	Avg.	MNLI Acc.	QNLI Acc.	QQP F1 / Acc.	SST-2 Acc.
Single Task	88.5	83.9, 84.2	90.9	87.7 / 90.9	93.2
Multi-Task	86.8	83.7, 83.7	90.5	86.3 / 89.6	
+ SST-2	87.7	83.2, 83.3	90.8	86.8 / 90.2	92.0
TKD $_{\text{MNLI + QNLI + QQP}}$	85.1	79.7, 80.1	87.9	87.1 / 90.5	
+ SST-2	85.0	79.3, 79.1	86.6	85.3 / 88.3	91.5
UKD $_{\text{MNLI + QNLI + QQP}}$	**86.7**	83.1, 83.3	90.2	86.7 / 90.0	
+ SST-2	**87.0**	82.6, 82.6	90.2	83.4 / 90.8	92.5
TKD $_{\text{MNLI+ QQP + QNLI}}$	84.7	80.8, 80.7	90.2	83.1 / 88.5	
+ SST-2	80.7	76.8, 76.4	89.1	69.0 / 82.0	90.8
UKD $_{\text{MNLI + QQP + QNLI}}$	**86.8**	83.2, 83.3	89.8	87.1 / 90.5	
+ SST-2	**87.3**	83.1, 83.1	89.4	86.0 / 90.0	92.0
TKD $_{\text{QQP + MNLI + QNLI}}$	83.8	81.0, 81.9	90.1	79.5 / 86.5	
+ SST-2	77.3	67.7, 69.1	88.2	66.6 / 80.8	91.5
UKD $_{\text{QQP + MNLI + QNLI}}$	**86.5**	82.6, 83.0	90.0	86.7 / 90.3	
+ SST-2	**86.6**	82.3, 81.7	89.3	85.1 / 89.5	92.0
TKD $_{\text{QQP + QNLI + MNLI}}$	85.4	83.3, 83.2	88.1	83.9 / 88.4	
+ SST-2	84.2	82.1, 82.2	87.5	76.6 / 84.8	92.1
UKD $_{\text{QQP + QNLI + MNLI}}$	**86.3**	82.8, 83.3	89.1	86.5 / 90.0	
+ SST-2	**86.5**	82.2, 82.4	88.2	84.6 / 89.2	92.2
TKD $_{\text{QNLI + QQP + MNLI}}$	86.0	82.4, 83.1	88.4	86.2 / 89.8	
+ SST-2	85.5	74.6, 76.4	97.5	84.6 / 87.9	92.0
UKD $_{\text{QNLI + QQP + MNLI}}$	**86.6**	83.2, 82.8	90.1	86.6 / 90.1	
+ SST-2	**86.9**	83.1, 82.5	89.9	84.6 / 89.3	91.7
TKD $_{\text{QNLI + MNLI + QQP}}$	85.9	81.2, 81.5	89.5	86.9 / 90.3	
+ SST-2	83.4	77.8, 77.6	89.1	77.7 / 85.7	92.2
UKD $_{\text{QNLI + MNLI + QQP}}$	**87.0**	83.3, 83.4	90.4	87.3 / 90.5	
+ SST-2	**87.2**	83.0, 82.6	90.2	85.9 / 89.9	91.7

Table 3: TKD and UKD performances when incrementally learning three and four tasks. T_1 + T_2 + T_3 means pre-trained BERT was fine-tuned for T_1 first. Then T_2 and T_3 were added incrementally. +SST-2 rows show the result after further adding SST-2.

4.4 Adding Third and Fourth Tasks

We further explore the effectiveness of UKD by incrementally learning three and four tasks, and we report the results with different task orders in Table 3. Observing the weak performance of the OL, EM, and EWC systems for learning the second task, we exclude these baselines for the further experiments as adding more tasks can only degrade the performance more. Results show that UKD is able to provide useful information to retain the knowledge in the model. For instance, when adding MNLI and QNLI to QQP, the F1 score of QQP drops about 8% with TKD, while using UKD the drop is only about 1% compared to the single task model. Notice that this pattern is consistent in almost every task combination we experimented with.

To further confirm the capability of the UKD approach to deal with this setting, we add a very different fourth task, i.e., SST-2. The main difference of this task with MNLI, QNLI and QQP stems from the fact that SST-2 is a single sentence classification task while the others are sentence pair tasks. Again, using UKD minimizes the forgetting (e.g., 2.6% drop for QQP vs. 21.1% drop with TKD). Such improvement is due to the usage of the unlabeled data for the old tasks; without such data the model observes a very different data distribution as the SST-2 task is very different from the other three.

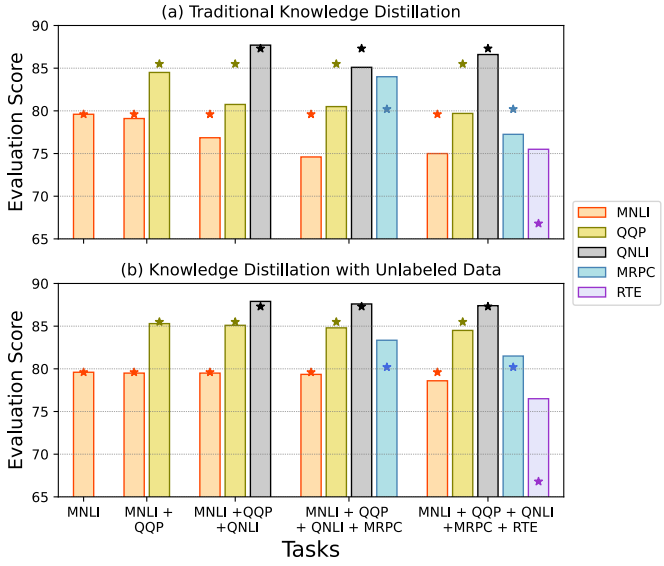

Figure 2: After training an MNLI model, we incrementally add QQP, QNLI, MRPC, and RTE. '⋆' markers represent the ST fine-tuning performance for the corresponding task.

4.5 Incremental Addition of Five Tasks

So far, we used identical unlabeled data for UKD at each step. We now simulate the scenario in which we cannot store any data. We report a continual learning experiment on five different tasks with the order[3] MNLI → QQP → QNLI → MRPC → RTE, where the unlabeled data is different at every step of UKD.

We divide each training set in equal slices and we use one as labeled data and the others as new unlabeled data in future training steps. For example, the first task (MNLI) is divided in five slices, the second task (QQP) in four slices, and so on. At each step, we train the model on one of the slices of the task we want to add, and we employ an unused slice of each previous task as unlabeled data. For instance, in the first step we simply train a model on one slice of MNLI; in the second stage, we use a different slice of MNLI as unlabeled data for the UKD approach and the first slice of QQP as annotated data to learn the new task, and so on. At the end, we'll have used all the data without ever observing the same example in more than one stage.

Figure 2 shows that UKD also outperforms TKD in this setting.[4] Incrementally adding a new task contributes to the forgetting of older tasks for TKD. For example, MNLI performance drops at each step, resulting, at the last stage, in a total drop of about 5% drop in accuracy (Figure 2a). However, with UKD we can observe that our approach is able to maintain the performance for old tasks (Figure 2b). For instance, the performance for the first task (MNLI) is almost

[3]Tasks order reflects the inverse order of the dataset sizes to have a reasonable amount of training/distillation data at each step.
[4]As RTE is smaller than the other datasets, we up-sampled (4x) the training data.

the same at each step. This experiment demonstrates that the usage of data from the same data distribution of the old tasks is beneficial for avoiding the catastrophic forgetting. Another interesting insight is that when adding new tasks to a model already exposed to several fine-tuning stages, the model is still able to achieve good results on the new task. This means that fine-tuning with our strategy does not affect the general knowledge learned in the language model pre-training. Given that the knowledge transfer effect may be limited, as indicated by the TKD results, we argue that the our approach is limiting the loss of the general linguistic knowledge learned in the language model pre-training. Therefore, our proposed approach is feasible for real-world applications like voice assistants where new tasks with labeled training data could be added to a pool of existing tasks in a model.

In Table 4 we report the results of an additional task ordering, i.e., QNLI → QQP → MNLI → RTE → MRPC. We observe that despite changing the order of the task, the outcome is the same. We observed the similar pattern when we experimented with another task order different than the mentioned ones. Our proposed model is able to limit the catastrophic forgetting happening with other solutions in a continual learning setting.

	Avg.	QNLI	QQP	MNLI	RTE	MRPC
		Acc.	F1 / Acc.	Acc.	Acc.	F1 / Acc.
Single Task	81.8	86.8	83.6 / 87.4	81.0, 81.6	64.3	87.9 / 82.1
Step 2: QQP added to QNLI						
MTL	85.9	87.1	83.2 / 87.3			
TKD	86.1	87.2	83.5 / 87.7			
UKD	86.4	87.7	83.6 / 87.9			
Step 3: MNLI added to [QNLI, QQP]						
MTL	83.8	87.5	83.2 / 87.0	80.0, 81.2		
TKD	83.4	85.9	82.6 / 87.0	80.3, 81.0		
UKD	84.0	87.8	83.2 / 87.6	80.4, 81.1		
Step 4: RTE added to [QNLI, QQP, MNLI]						
MTL	81.9	86.8	83.4 / 87.2	79.6, 80.1	74.4	
TKD	80.1	84.9	81.1 / 85.5	75.8, 76.3	76.9	
UKD	82.4	85.7	84.6 / 86.7	79.9, 80.0	77.2	
Step 5: MRPC added to [QNLI, QQP, MNLI, RTE]						
MTL	82.0	87.4	83.1 / 87.0	80.3, 81.0	74	84.8 / 78.7
TKD	79.9	85.6	80.7 / 86.2	76.2, 77.3	60.3	88.7 / 84.1
UKD	82.5	87.3	82.6 / 87.0	79.2, 80.2	74.7	86.9 / 81.9

Table 4: Results of incrementally learning five tasks. We first fine-tune a pre-trained BERT model on QNLI. Then we incrementally add QQP, MNLI, RTE, and MRPC to that model.

4.6 Continual Learning in Real-World Applications

To further validate the assumption that our method can be useful in practical settings, we conduct experiments on a dataset built from traffic directed to a leading digital voice assistant. We tackle the task of Intent Classification with respect to 5 different domains, each containing numerous intents. More specifically, we perform experiments where we add new domains over time. This realistically mimics a practical scenario where voice assistants are dynamically updated to support new functionalities in new domains.

Domain	Intents	Train	Dev	Test	Skewness
D1	12	1901	407	408	0.01
D2	11	1311	280	282	2.09
D3	5	995	213	214	1.49
D4	11	611	131	132	2.77
D5	13	396	85	86	2.96

Table 5: Statistics of the voice assistant data. There are multiple intents for each domain and data distribution is generally skewed towards the majority class for most domains.

In Table 5 we report the statistics of the dataset we built with respect to the different domains. The amount of training data in this setting is quite limited, and the number of categories intents for each domain is between 5 and 13.

In Table 6 we show the results of the Intent Classification on the five different domains by incrementally adding one domain to a pre-trained large language model. Again, we adopted the BERT-base-uncased model, fine-tuned for 20 epochs with early stopping. We adopted the same hyper-parameters as in the previous experiments, except for the batch size. Given the limited data, we adopted a batch size of 16. For reasons of confidentiality, we report the relative performance compared to the multi-task setting. As in the previous experiments, the MTL performance can be considered as an upper bound. The results show that every task gets little to large performance gain by the traditional multi-task training with respect to a ST fine-tuning (first two rows of Table 6). This is intuitive as in the MTL training the model can exploit information shared across tasks. The gain is high for the domains with smaller datasets. For example, D5, which is the least represented domain, gets a boost of 30 points with respect to ST.

	D1	D2	D3	D4	D5	Avg.
Baseline (MTL)	0	0	0	0	0	0
ST	-0.67	-1.28	-0.79	-21.35	-30.58	-10.93
TKD $_{[D1,D2]}$	-2.12	-0.80				**-1.46**
UKD $_{[D1,D2]}$	-2.31	-1.93				-2.12
TKD $_{[D1,D2,D3]}$	-4.89	-4.08	-3.17			-4.05
UKD $_{[D1,D2,D3]}$	-4.05	-5.19	-1.68			**-3.64**
TKD $_{[D1,D2,D3,D4]}$	-12.9	-5.52	-5.39	-1.55		-6.34
UKD $_{[D1,D2,D3,D4]}$	-3.43	-1.59	-3.81	-5.61		**-3.61**
TKD $_{[D1,D2,D3,D4,D5]}$	-17.74	-7.84	-7.96	-1.34	-26.9	-12.36
UKD $_{[D1,D2,D3,D4,D5]}$	-2.4	-0.74	-3.95	-5.64	-3.07	**-3.16**

Table 6: Incrementally adding Intent Classification tasks for five different domains. We report the relative F1 score change considering the multi-task setting as the baseline, i.e., a negative value indicates a drop from the multi-task baseline. For example, single task fine-tuning on D1 achieves 0.67 less F1 score compared to multi-task training.

In consecutive rows, we observe that, when we add D2 to a model trained only on D1, TKD performs slightly better than UKD. However, when we add more tasks, UKD starts outperforming TKD. For example, adding D3 and D4 causes TKD to drop of about 12.9 and 5.5 points with respect to the first task and second domains. Instead, adding D3 with UKD results in a drop of only 3.43 and 1.59 for the two domains, respectively. Sometimes the performance of the newest task is better of TKD than UKD (e.g., when adding D4, TKD drop with respect to MTL is 1.55 while UKD drop is 5.61). As we have the control over the data of newly added tasks, we can apply techniques like oversampling or hyper-parameter tuning to improve the new task's performance. In general, the results obtained with UKD seems more promising, as the average drop over all the steps for all the domains with respect to the MTL baseline is -3.42 for UKD while it is -7.3 for TKD.

5 CONCLUSION

In this paper, we proposed a semi-supervised approach based on the Knowledge Distillation framework to incrementally learn new natural language tasks in already trained models. We concentrated our analysis on maintaining a consistent performance on the tasks already learned, i.e., to prevent the catastrophic forgetting of the model. We suggest the usage of an unlabeled set of data coming from the same data distribution of the previously acquired tasks to stabilize the training in the knowledge distillation framework. Experimental results on both publicly available benchmarks and a dataset we built from real voice-assistant data demonstrate that the usage of unlabeled data from previous task distribution is crucial to prevent the catastrophic forgetting phenomenon when dealing with different data distributions.

We expect this technique to be useful for many practical problems when a model must be updated over time to support new tasks. This is a common scenario in industry settings, where systems are updated over time to support new functionalities. In the future, it may be interesting to study the problem of automatically generating data representing the old task data distributions starting from the already trained models.

REFERENCES

[1] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. 2021. Rainbow Memory: Continual Learning with a Memory of Diverse Samples. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 8214–8223. https://doi.org/10.1109/CVPR46437.2021.00812

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

[3] Giuseppe Castellucci, Simone Filice, Danilo Croce, and Roberto Basili. 2021. Learning to Solve NLP Tasks in an Incremental Number of Languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, Online, 837–847. https://doi.org/10.18653/v1/2021.acl-short.106

[4] Zhiyuan Chen, Bing Liu, Ronald Brachman, Peter Stone, and Francesca Rossi. 2018. *Lifelong Machine Learning* (2nd ed.). Morgan-Claypool Publishers.

[5] Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D. Manning, and Quoc V. Le. 2019. BAM! Born-Again Multi-Task Networks for Natural Language Understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 5931–5937. https://doi.org/10.18653/v1/P19-1595

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[7] William B Dolan and Chris Brockett. 2005. Automatically Constructing a Corpus of Sentential Paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

[8] Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. arXiv:1312.6211 [stat.ML]

[9] Jiangpeng He and Feng Zhu. 2021. Unsupervised Continual Learning Via Pseudo Labels. *ArXiv* abs/2104.07164 (2021).

[10] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*. http://arxiv.org/abs/1503.02531

[11] Yufan Huang, Yanzhe Zhang, Jiaao Chen, Xuezhi Wang, and Diyi Yang. 2021. Continual Learning for Text Classification with Information Disentanglement Based Regularization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 2736–2746. https://doi.org/10.18653/v1/2021.naacl-main.218

[12] Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. First Quora Dataset release: Question Pairs. *data.quora.com* (2017).

[13] Xisen Jin, Dejiao Zhang, Henghui Zhu, Wei Xiao, Shang-Wen Li, Xiaokai Wei, Andrew O. Arnold, and Xiang Ren. 2021. Lifelong Pretraining: Continually Adapting Language Models to Emerging Corpora. *ArXiv* abs/2110.08534 (2021).

[14] Zixuan Ke, Bing Liu, and Xingchang Huang. 2020. Continual Learning of a Mixed Sequence of Similar and Dissimilar Tasks. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 18493–18504. https://proceedings.neurips.cc/paper/2020/file/d7488039246a405baf6a7cbc3613a56f-Paper.pdf

[15] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences* 114, 13 (2017), 3521–3526. https://doi.org/10.1073/pnas.1611835114 arXiv:https://www.pnas.org/content/114/13/3521.full.pdf

[16] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2019. Continual learning: A comparative study on how to defy forgetting in classification tasks. arXiv:1909.08383 [cs.CV]

[17] Timothée Lesort, Massimo Caccia, and Irina Rish. 2021. Understanding Continual Learning Settings with Data Distribution Drift Analysis. *ArXiv* abs/2104.01678 (2021).

[18] Yuanpeng Li, Liang Zhao, Kenneth Church, and Mohamed Elhoseiny. 2020. Compositional Language Continual Learning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rklnDgHtDS

[19] Z. Li and D. Hoiem. 2018. Learning without Forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 12 (2018), 2935–2947.

[20] Tianlin Liu, Lyle Ungar, and João Sedoc. 2019. Continual Learning for Sentence Representations Using Conceptors. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 3274–3279. https://doi.org/10.18653/v1/N19-1331

[21] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-Task Deep Neural Networks for Natural Language Understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 4487–4496. https://doi.org/10.18653/v1/P19-1441

[22] Michael McCloskey and Neal J Cohen. 1989. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. *Psychology of Learning and Motivation - Advances in Research and Theory* 24, C (1 jan 1989), 109–165. https://doi.org/10.1016/S0079-7421(08)60536-8

[23] Natawut Monaikul, Giuseppe Castellucci, Simone Filice, and O. Rokhlenko. 2021. Continual Learning for Named Entity Recognition. In *AAAI*.

[24] Shiva Pentyala, Mengwen Liu, and Markus Dreyer. 2019. Multi-Task Networks with Universe, Group, and Task Feature Learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 820–830. https://doi.org/10.18653/v1/P19-1079

[25] Amal Rannen, Rahaf Aljundi, Matthew B. Blaschko, and Tinne Tuytelaars. 2017. Encoder Based Lifelong Learning. In *The IEEE International Conference on Computer Vision (ICCV)*.

[26] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. 2017. iCaRL: Incremental Classifier and Representation Learning. In *CVPR*.

[27] Sebastian Ruder. 2017. An Overview of Multi-Task Learning in Deep Neural Networks. *CoRR* abs/1706.05098 (2017). arXiv:1706.05098 http://arxiv.org/abs/

1706.05098

[28] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive Neural Networks. arXiv:1606.04671 [cs.LG]

[29] Yilin Shen, Xiangyu Zeng, and Hongxia Jin. 2019. A Progressive Model to Enable Continual Learning for Semantic Slot Filling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 1279–1284. https://doi.org/10.18653/v1/D19-1126

[30] K. Shmelkov, C. Schmid, and K. Alahari. 2017. Incremental Learning of Object Detectors without Catastrophic Forgetting. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 3420–3429.

[31] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, 1631–1642. https://www.aclweb.org/anthology/D13-1170

[32] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F. Grewe. 2020. Continual learning with hypernetworks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SJgwNerKvB

[33] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=rJ4km2R5t7

[34] Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. 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)*. 1112–1122.

[35] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv* abs/1910.03771 (2019).