Unsupervised Lifelong Learning with Curricula

Anonymous Author(s)
Submission Id: 335

ABSTRACT

Lifelong machine learning (LML) has extensively driven the development of web applications, enabling the learning systems deployed on web servers to deal with a sequence of tasks in an incremental fashion. Such systems can retain knowledge from learned tasks in a knowledge base and seamlessly applying it to improve the future learning. Unfortunately, most existing LML methods require labels in every task, whereas providing persistent human labeling for all future tasks is costly, onerous, error-prone, and hence impractical. Motivated by this situation, we propose a new paradigm named *unsupervised lifelong learning with curricula* (ULLC), where only one task needs to be labeled for initialization and the system then performs lifelong learning for subsequent tasks in an *unsupervised* fashion. A main challenge of realizing this paradigm lies in the occurrence of negative knowledge transfer, where partial old knowledge becomes detrimental for learning a given task yet cannot be filtered out by the learner without the help of labels. To overcome this challenge, we draw insights from the learning behaviors of humans. When faced with a difficult task that cannot be well tackled by our current knowledge, we usually postpone it and work on some easier tasks first, which allows us to grow our knowledge. Thereafter, once we go back to the postponed task, we are more likely to tackle it well as we are more knowledgeable now. The key idea of ULLC is similar – at any time, a pool of candidate tasks are organized in a *curriculum* by their distances to the knowledge base. The learner then starts from the closer tasks, accumulates knowledge from learning them, and moves to learn the faraway tasks with a gradually augmented knowledge base. The viability and effectiveness of our proposal are substantiated through both theoretical analyses and empirical studies.

1 INTRODUCTION

Machine learning has been instrumented in developing models for advancing web search and knowledge mining [11, 12, 27]. These models are usually developed in an *isolated* paradigm, where each model is trained on a dataset drawn for solving a specific learning task only. Once the task shifts or a different task arrives, another dataset needs to be collected and manually labeled, on which a new model is trained for the shifted or new task. A comparison with human intelligence reveals the inefficiency of this isolated machine learning paradigm. As a matter of fact, we humans rarely learn in isolation; Instead, we retain knowledge from what we have learned, so as to become more competent in future problem solving. Consider especially that the datasets in web applications are usually very large and constantly skewed, a repeated process of data collecting and new model training is rather inefficient, where possible knowledge reuse is ignored.

Lifelong machine learning (LML), mimicking such a human learning ability, has been proposed to address the shortcomings of the isolated learning paradigm [24, 38, 47]. In particular, LML maintains a *knowledge base*, which accumulates the knowledge learned from the tasks seen so far. Given a sequence of disparate tasks, the goal of LML is to maximize the prediction performance of each arriving task by leveraging the knowledge base. So far, LML has promoted a wide range of new algorithms and systems in real applications on web, such as semantic search [10], opinion mining [51], internet commerce [5], and recommender systems [41], among many others.

Despite effective, existing LML learners highly rely on labels scattering in all tasks to achieve good performance. Once no label exists in an arriving task, the learner cannot identify which pieces of previously learned knowledge are applicable for reuse and which are detrimental. To illustrate, consider a learner for sentiment classification. Assume, its first task is to classify the movie reviews of "Superman 4 (1987)" into positive or negative sentiments. After reading a review, *e.g.*, "The visual effect sucks, the moon is like a toy", a piece of knowledge that the word "toy" indicates a negative opinion is learned and stored in the knowledge base. However, when the learner is used to classify a new movie (task), *e.g.*, "Toy Story (1995)", the word "toy" does not indicate any sentimental meanings. With no label in this new task, the pieces of detrimental knowledge are forcibly transferred from the knowledge base to it, leading to substantial prediction errors. Even worse, as lifelong learning can be deemed as an online bootstrapping process [38], these errors will be propagated and escalated to subsequent tasks to generate more errors. The overall learning performance could thus be significantly deteriorated.

To avert error propagation and escalation (a result of detrimental knowledge transfer), labels are required. Unfortunately, requiring all tasks being labeled is overwhelming. In practice, labels are often not available for various reasons. At the user end, for example, providing labels is in general costly, onerous, and error-prone. At the system end, tasks have different priorities while some tasks are instantaneous (*e.g.*, terrorism detection in social networks [48]), making planning and pre-labeling close to impossible.

Motivated by this situation, this work investigates an important question: Can an LML system, after being trained on one single labeled task, continually learn from subsequent tasks in an *unsupervised* fashion? To answer this question, we draw insight from human learning instinction. Rather than learning future tasks in an arbitrary order, humans usually organize tasks in a meaningful *curriculum*, starting from tasks that they are more familiar with and gradually moving to unfamiliar ones. During this process, as

WWW '21, April, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

familiar tasks can be learned in high confidence even without supervision, humans are likely to become more knowledgeable for dealing with those originally unfamiliar tasks by forming a knowledge base without inducing many errors.

We cast this insight into a novel LML paradigm, termed as *unsupervised lifelong learning with curricula* (ULLC). The key idea is, instead of passively receiving and learning tasks in a random sequence, ULLC actively chooses the next task to learn by organizing a few future tasks into a curriculum. High-level design of ULLC is as follows. At the beginning, a number of tasks are pooled in a buffer, and the knowledge base is initialized as the originally labeled task. After that, candidate tasks in the buffer are ordered in a curriculum according to their distances to the knowledge base. The learner chooses the closest task to learn and predicts the instances in this task. The labeled instances are ranked by their prediction confidence, measured by, *e.g.*, the margin-maximization principle [57], and those instances predicted in high confidence level will be merged into the knowledge base for knowledge accumulation. When a new task arrives, it is added to the buffer, substituting the learned task. The curriculum is updated by reordering the candidate tasks, respecting the knowledge base that now carries new instances. This process continues until no more future task arrives.

A main challenge in realizing our paradigm lies in how to measure the distance between a candidate task and the knowledge base for long time spans. Choosing a distance metric in a priori is unrealistic, since a fixed metric may work well at the beginning but can fail later on. For example, a metric posits the data distribution underlying tasks follow Gaussians [55, 60] may incur errors when the assumption does not hold in future tasks.

To support precise curriculum design, we tailor a new distance metric. Our metric is attained by calibrating the *complexity* of the feature alignment function that extracts the domain-invariant, common features between a candidate task and the knowledge base. Intuitively, the more faraway the two domains are, the more difficult the common features can be extracted, and hence the more complex the feature alignment function would be. This intuition lends us to a novel *elastic domain adversarial network* (EDAN) design, whose depth is adaptive and will be learned per task, approximating feature alignment functions at different complexity levels. Those complexity levels hence serve as a metric for ordering the candidate tasks by their distance to the knowledge base. Note, this new metric does not impose any assumption on data distribution or task structure, it is thereby likely to be adaptable for long time spans.

Specific contributions in this paper are summarized as follows.

(1) We propose a new ULLC paradigm that performs lifelong learning with a one-time labeling effort only. The fact that our paradigm can learn all future tasks in an *unsupervised* fashion is especially promising and more applicable than prior works that require full labeling information in all tasks.
(2) We devise a novel EDAN for adaptive distance metric, helping order the candidate tasks in a curriculum by their distances to the knowledge base with high precision and flexibility. A theoretical analysis shows that our curriculum learning strategy can provably lead to performance improvement over task learning in an arbitrary order.
(3) We have carried out extensive experiments over both synthetic and real datasets. Empirical results show that our approach can effectively overcome the detrimental knowledge transferring issue in an unsupervised setting, and its performance is comparable to supervised learning competitors.

The rest of this paper proceeds as follows. Section 2 reviews related work. Section 3 formalizes our learning problem, spotlights the challenge, and unfolds the high-level idea of our design. Section 4 elaborates the proposed approach. Section 5 presents theoretical analysis. Section 6 reports experimental results and Section 7 concludes the work. Due to space limitation, proofs and derivation details are deferred to supplemental material (https://bit.ly/2T6nvJB).

2 RELATED WORK

Our ULLC paradigm is closely related with Lifelong Learning, Multitask Learning, and Domain Adaption. In this section, we review the prior literatures in the three research avenues and discuss the relations and differences between our approach and theirs.

Lifelong Learning, *a.k.a.* Continual Learning [29, 32, 42] or Never-Ending Learning [1, 23, 36], aims to build general-purpose machines that can learn from incrementally more tasks after being initially trained. The crux of this research line lies in the overcoming of catastrophic forgetting, *i.e.*, the loss or disruption of previously learned knowledge when new knowledge is added. In general, existing methods fall into two categories. One category comprises the model-based methods, where the model parameters are regularized to avoid drastic updates, striving to search a Pareto-effective solution that performs satisfactorily for all seen tasks [3, 29, 32, 46]. The other category covers the rehearsal-based methods, where the historical instances are (partially) stored in an external memory (*i.e.*, the knowledge base) and will be jointly trained along with new tasks [18, 34, 40, 42, 45]. Unfortunately, existing methods mostly require full knowledge of task labels whereas proving such continuous human supervision in all future tasks is unrealistic or too expensive, hindering their deployment in real practices. Our approach much lifts this assumption, entailing a one-time labeling effort in a single task only and envisioning no label from future tasks, thereby enjoying a broader applicability.

Multitask Learning explores potential synergies across a set of learning tasks in which each task suffers from insufficient training instances. Prior studies have delivered both theoretical insights [4, 7, 22] and empirical evidences [2, 21, 61] to show that, if the multiple tasks are truly related, then the knowledge in one task can guide the learning of other tasks, such that the sample complexity of all learning tasks can be improved through jointly training. Once the relatedness among tasks is weak, knowing which piece of knowledge is shareable becomes important [26] because the knowledge of one task could be irrelevant or adversarial to other tasks, which is somehow close to the idea of combatting detrimental/negative knowledge transfer in our context. However, the existing works prescribe all tasks to be available beforehand, which are prohibitive and inflexible in the sense that they do not support learning in an on-line process. As new tasks arrive, their learning systems are retrained from sketch by scanning both old and new data in multiple iterations, leading to both memory and computational overheads. Our ULLC paradigm does not bother to store a massive

volume of data from multiple tasks before learning begins; Instead, it allows the learning on the go, thereby being more flexible and computational and memory friendly than multitask learning.

Domain Adaption, *a.k.a.* transfer learning [20, 37, 50] or transductive learning [28, 39, 49, 62], strives to improve the learning efficiency of one label-scarce domain as a target with the help of one or multiple label-rich domains as auxiliaries. The key technique is to extract a set of latent, domain-invariant features as a bridge, through which the auxiliary-trained learners can be propagated to the target domain. If the target domain is totally unlabeled, the learning problem upgrades to unsupervised domain adaption (UDA) [17, 33, 59], which is more challenging in the sense that no target label is available for examining the existence of negative knowledge transfer, where, in their context, the target domain shifts and hence follows a quite disparate distribution from the auxiliaries. Pioneer studies [8, 53, 58] circumvent negative transfer by filtering out unrelated auxiliary data, which can only be realized under two restrictive assumptions. First, they require all auxiliary domains to be labeled with groundtruth and readily available in a batch, so they cannot work well in our setting where the tasks arrive in sequence and only one (*i.e.*, the first) task is labeled. Second, they posit a fixed distance metric that will be valid for all domains in future, which is less generalizable to data that do not follow the prescribed distributions, *e.g.*, Gaussians [55, 60]. Our ULLC paradigm does not make these assumptions and thus is more general. Moreover, our approach focuses on the improvements to all seen tasks by maintaining a knowledge base, rather than their methods that concern accurate modeling in the target domain only.

3 THE ULLC PARADIGM

Given a sequence of tasks $\{\mathcal{T}_i \mid i = 0, 1, \ldots, N\}$ in which we suppose, without loss of generality, that $\mathcal{T}_0$ is with labels and all other tasks $\{\mathcal{T}_i\}_{i=1}^{N}$ remain unlabeled. Let $\mathrm{P}_{\mathcal{T}_0}(X, Y)$ and $\mathrm{P}_{\mathcal{T}_i}(X)$ denote the instance-label joint distribution of $\mathcal{T}_0$ and the marginal distribution of $\mathcal{T}_i$, respectively, with X being the random variable in an $\mathbb{R}^d$ input space and Y being the classification label. At each time step, one task is learned with its data instances predicted (labeled). A knowledge base $\mathrm{R}^{(i)}$ is maintained that retains and accumulates the knowledge (represented by the labeled instances) from the perviously learned tasks, *i.e.*, $\{\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_{i-1}\}$.

3.1 The Learning Problem

Our goal is to learn a a series of good hypotheses $h_1, \ldots, h_N \in \mathcal{H}$, with $\mathcal{H}$ being a hypothesis space, that makes accurate prediction for the unlabeled tasks $\mathcal{T}_1, \ldots, \mathcal{T}_N$. The hope is that the knowledge base can always provide useful knowledge for any given task, such that a hypothesis learned on $\mathrm{R}^{(i)}$ can be transferred to a newly arriving task $\mathcal{T}_i$ seamlessly, achieving decent classification performance.

To achieve this goal, we use feature alignment [17], where the key technique is to extract a common space that closes the cross-domain discrepancy between $\mathrm{R}^{(i)}$ and any given $\mathcal{T}_i$. Specifically, a feature alignment function is a mapping $\phi : \mathbb{R}^d \mapsto \mathbb{R}^z$, where the z-dimensional latent space is spanned by a set of task-invariant common features. As such, the data from $\mathrm{R}^{(i)}$ and any $\mathcal{T}_i$, after being mapped via ϕ, would follow a similar marginal distribution, *i.e.*, $\mathrm{P}(\phi(\mathrm{X}_{\mathrm{R}^{(i)}})) \approx \mathrm{P}(\phi(\mathrm{X}_{\mathcal{T}_i}))$. A hypothesis space defined on this latent space can hence yield a desired hypothesis h_i that works well on both $\mathrm{R}^{(i)}$ and $\mathcal{T}_i$. The hypothesis h_i and the feature alignment function ϕ are jointly trained through playing a min-max game as:

$$\min_{h} \max_{\phi} \mathbb{E}_{\mathrm{x}, y \sim \mathrm{R}^{(i)}} \big[y \neq h_i(\phi(\mathrm{x})) \big] - \mathrm{D}\big[\mathrm{P}(\phi(\mathrm{X}_{\mathrm{R}^{(i)}})) \,\big\|\, \mathrm{P}(\phi(\mathrm{X}_{\mathcal{T}_i})) \big], \tag{1}$$

where the first term represents the empirical risk suffered by predicting labels for data in $\mathrm{R}^{(i)}$ and the second term calibrates the cross-domain distributional divergence between $\mathrm{R}^{(i)}$ and $\mathcal{T}_i$.

3.2 Challenge: Negative Knowledge Transfer

Each time step, a task $\mathcal{T}_i$ arrives, in which, unfortunately, its data may follow an extensively disparate distribution from those stored in $\mathrm{R}^{(i)}$. Extracting the common space between two highly disparate domains is fundamentally difficult, so a very complex mapping ϕ is required, such that the second term in Eq. (1) can be minimized. For such cases, deep neural networks [17, 33] are widely-used function approximators that can learn ϕ at arbitrary complexity levels.

However, an overly complex ϕ is very likely to overfit h_i to the data in $\mathrm{R}^{(i)}$ and yield inferior prediction performance on $\mathcal{T}_i$. The main reason is that those pieces of knowledge in $\mathrm{R}^{(i)}$, which are detrimental (negative) for learning $\mathcal{T}_i$, are included in optimizing Eq. (1), incurring a phenomenon known as *negative knowledge transfer* [8, 53]. Such negative knowledge is non-detectable as no label is available in $\mathcal{T}_i$. We extrapolate the reasons as follows.

As the hypothesis h_i is defined over the latent representations $\phi(\mathrm{x})$, we can deem the entire function $h_i(\phi(\cdot))$ as the predictor, which takes as input the data instances and outputs the predicted labels. To fullfil a large cross-domain gap, the mapping ϕ needs to be overly complex, so do the predictor $h_i(\phi(\cdot))$. The more complex the predictor, the better it fits to the training data (*i.e.*, the labeled data in $\mathrm{R}^{(i)}$ in our context), and the worse it generalizes to the unseen data (*i.e.*, the unlabeled data in $\mathcal{T}_i$). As a result, if no label is available in $\mathcal{T}_i$, this overfitting cannot be detected and alerted, such that substantial prediction errors in $\mathcal{T}_i$ will be included in $\mathrm{R}^{(i)}$ with wrongly labeled instances. Even worse is that these errors will propagate and escalate in learning the subsequent tasks, as in an online bootstrapping process [38], where more errors will be generated and accumulated. The overall learning performance is hence much deteriorated, necessitating the combatting against the negative knowledge transfer challenge.

3.3 Our Idea: Knowledge Retention with Curriculum Learning

To overcome negative knowledge transfer, we abstract the human learning intinction into an inductive bias. Like humans who usually start with tasks that they are familiar with, our learner orders tasks in a curriculum, starts learning the task that shares the most commonality with the data in $\mathrm{R}^{(i)}$, grows knowledge from them, and gradually become knowledgeable for dealing with those disparate tasks. Through this way, since the task learned at each time step can be accurately predicted even without the labels, few prediction errors are accumulated. The overall classification performance across the subsequent tasks can hence be improved.

In particular, for the task sequence with a long time span, we pool arriving tasks into a buffer of size K and, at each time step, the

candidate tasks in the buffer are ordered by their distances to $\mathrm{R}^{(i)}$. We then select the task being closest to $\mathrm{R}^{(i)}$ and jointly learn the hypothesis h_i and the feature alignment function ϕ between the selected task and $\mathrm{R}^{(i)}$. After learning, we predict the data in the selected task, and thereafter the instances being predicted with the highest confidence level are merged into $\mathrm{R}^{(i)}$. At the next round, a new task is buffered and all candidate tasks are re-ordered in a new curriculum to respect the new instances added in $\mathrm{R}^{(i)}$. This process continues until no more task arrives.

4 OUR APPROACH

To implement the high-level idea into a concrete algorithm, the crux of our approach lies in how to order the candidate tasks into a meaningful curriculum. Section 4.1 and Section 4.2 serve to unfold the technical details of our curriculum design. Specifically, Section 4.1 presents a novel elastic domain adversarial network (EDAN) architecture, which propagates the labels from the knowledge base to a given task through adversarial training. Section 4.2 elaborates an adaptive domain-wise distance metric, which is derived from the trained EDAN and can support the designed curriculum with high flexibility and precision. After having the curriculum, we choose a task to learn. We end by exhibiting how the knowledge is retained by storing the most confidently predicted instances in Section 4.3.

4.1 Elastic Domain Adversarial Network

EDAN follows the spirit of pioneer works [17, 33] of exploiting a generative adversarial network (GAN) framework. Unlike existing methods that fix network depth in a priori, the main technical innovation of EDAN is to *treat the network depth as a learnable semantic*. Specifically, EDAN starts from an over-complete architecture, and it automatically decides how and when to adapt its network depth during the learning procedure, in accordance with the *complexity* level of the feature alignment function it needs to approximate. Figure 1 shows the computational graph of EDAN.

The key idea of EDAN is to extract the common space between the knowledge base $\mathrm{R}^{(i)}$ and an unlabeled new task $\mathcal{T}_i$ through adversarial training, where $\mathrm{R}^{(0)} := \mathcal{T}_0$ as an initialization. Given an instance x, we denote m as its task membership, indicating whether x is from $\mathrm{R}^{(i)}$ ($m = 0$ if $\mathrm{x} \sim \mathrm{R}^{(i)}$) or from $\mathcal{T}_i$ ($m = 1$ if $\mathrm{x} \sim \mathcal{T}_i$). For any instance from $\mathrm{R}^{(i)}$, its label is known, denoted by y.

Consider an over-complete network with L hidden layers, the output of its l-th hidden layer is recursively defined as:

$$\mathrm{h}_{(l)} = \mathcal{F}(\mathrm{h}_{(l-1)}; \mathrm{W}_{(l)}) = \sigma(\mathrm{W}_{(l)}^{\top}\mathrm{h}_{(l-1)}),$$
$$\forall l = 1, \ldots, L; \quad \mathrm{h}_{(0)} = \mathrm{x},$$

where $\mathcal{F}$ represents the feature alignment function for extracting the task-invariant latent features, parameterized by $\mathrm{W}_{(l)}$ and activated by a non-linear function $\sigma(\cdot)$ such as sigmoid, ReLU, *etc.*

Denoted by $\mathrm{h}_{(l)}$ the output of the l-th hidden layer. A classifier C and a task discriminator $\mathcal{D}$ predict the label and the task membership of $\mathrm{h}_{(l)}$ as $\hat{y}_{(l)} = C(\mathrm{h}_{(l)}; \theta_{y\,(l)})$ and $\hat{m}_{(l)} = \mathcal{D}(\mathrm{h}_{(l)}; \theta_{m\,(l)})$, respectively. EDAN linearly combines the sub-predictions suggested by all hidden layers to make the final predictions, namely, $\hat{y} = \sum_{l=1}^{L} \alpha_{(l)}\hat{y}_{(l)}$ and $\hat{m} = \sum_{l=1}^{L} \alpha_{(l)}\hat{m}_{(l)}$, where $\alpha_{(l)}$ denotes the weight factor of the l-th hidden layer. The objective of EDAN is defined by the following min-max game:

$$\min_{\mathcal{F},C} \max_{\mathcal{D}} \sum_{l=1}^{L} \alpha_{(l)} \Big(\mathcal{L}_{\mathrm{sup}}^{(l)}(\mathcal{F}, C) - \lambda \mathcal{L}_{\mathrm{adv}}^{(l)}(\mathcal{F}, \mathcal{D}) \Big), \tag{2}$$

$$\mathcal{L}_{\mathrm{sup}}^{(l)}(\mathcal{F}, C) = \mathbb{E}_{(\mathrm{x},y)\in \mathrm{R}^{(i)}}[\ell(y, \hat{y}_{(l)})], \tag{3}$$

$$\mathcal{L}_{\mathrm{adv}}^{(l)}(\mathcal{F}, \mathcal{D}) = \mathbb{E}_{(\mathrm{x},m)\in \mathrm{R}^{(i)}\cup\mathcal{T}_i}[\ell(m, \hat{m}_{(l)})], \tag{4}$$

where $\mathcal{L}_{\mathrm{sup}}^{(l)}(\mathcal{F}, C)$ and $\mathcal{L}_{\mathrm{adv}}^{(l)}(\mathcal{F}, \mathcal{D})$ represent the suffered supervised sub-loss and the adversarial sub-loss of the l-th hidden layer, respectively. Denoted by $\ell(\cdot, \cdot)$ a loss function, and λ a positive parameter to balance the two sub-loss terms.

The intuitions behind Eqs. (2) (3), and (4) are interpreted as follows. Each hidden layer extracts a common space that (i) represents the input x in a more separable form (minimizing the supervised loss) to satisfy the classifier; and (ii) closes the cross-task distribution discrepancy (maximizing the adversarial loss) to fool the discriminator. A good hidden layer should jointly incur small supervised loss and large adversarial loss. Thus, to optimize Eq. (2), we should increase weights for such good hidden layers and decrease weights for other layers. To do this, we update the weight factors using the hedging strategy [16, 44], defined as:

$$\alpha_{(l)} = \frac{\exp\left[-\tau \sum_{t=1}^{T} \left(\mathcal{L}_{\mathrm{sup}}^{(l)}(\mathcal{F}, C) - \lambda \mathcal{L}_{\mathrm{adv}}^{(l)}(\mathcal{F}, \mathcal{D})\right)\right]}{\sum_{l=1}^{L} \exp\left[-\tau \sum_{t=1}^{T} \left(\mathcal{L}_{\mathrm{sup}}^{(l)}(\mathcal{F}, C) - \lambda \mathcal{L}_{\mathrm{adv}}^{(l)}(\mathcal{F}, \mathcal{D})\right)\right]}, \tag{5}$$

which guarantees $\forall \alpha_{(l)} \in (0, 1)$. Denoted by τ the discount rate parameter, whose value assignment is discussed later in Theorem 5.1 of Section 5. The number of training iterations is represented by T.

Training EDAN is to searching parameters of $\mathcal{F}$, C, and $\mathcal{D}$ that deliver a saddle point of Eq. (2). In this work, we train EDAN with stochastic updates with backpropagation and gradient reversal operator [17]. Since the page limits preclude a detailed discussion, we defer the technical details of training EDAN to Section 1 of supplemental material.

4.2 Curriculum Design via EDAN

We now tailor a novel distance metric for ordering tasks in a curriculum. Specifically, among K candidate tasks in a buffer, we prioritize the task that shares the most common knowledge with the current base $\mathrm{R}^{(i)}$ at each time step. Our metric is to quantify this level of commonality between $\mathrm{R}^{(i)}$ and each candidate task.

For three reasons, the network depth of the learned EDAN, which is represented by the weight factors of the hidden layers, is a good device for this quantification. First, due to the diminishing feature reuse issue in overcomplete networks [25, 31], the deep layers in EDAN tend to converge slower than shallow layers. As a result, the output of shallow layers are likely to incur smaller overall loss (supervised sub-loss minus adversarial sub-loss), making weight factors of shallow layers larger than others according to Eq. (5).

Second, for converged layers, the prediction results of deep layers are more accurate than those of shallow layers, because the deep layers have larger learning capacities. Thus, over T training iterations, the deep layers in total suffer less loss than shallow layers, and thus the weight factors of deep layers are larger than those of shallow ones for such layers.

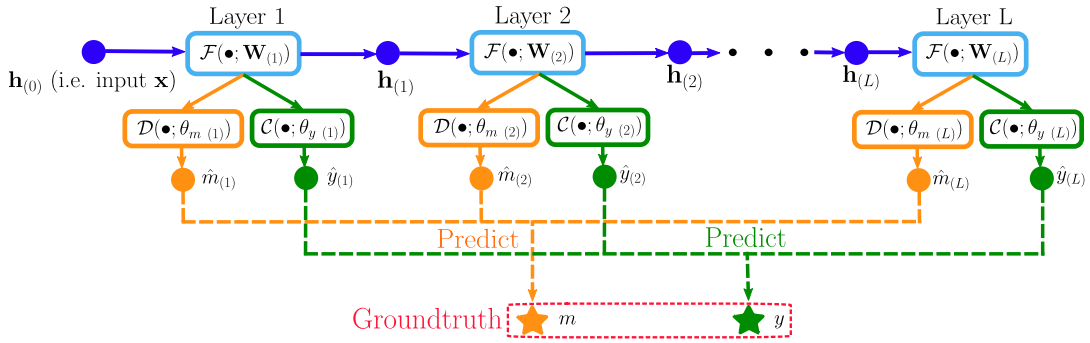
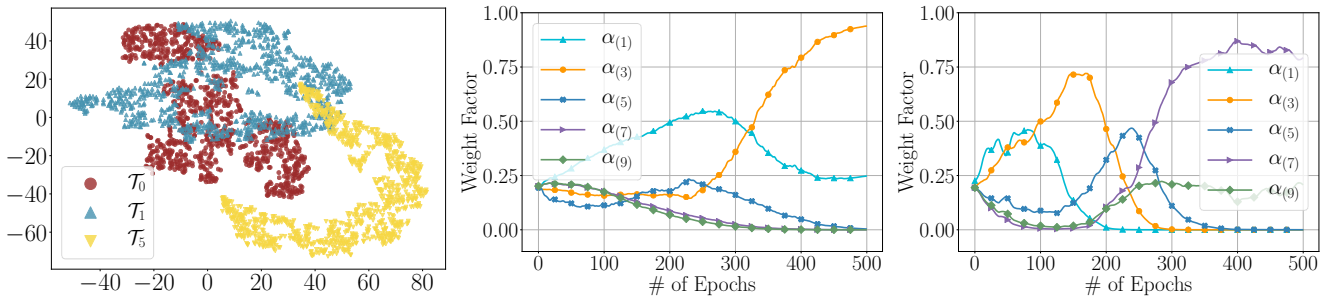


Figure 1: Computational graph of EDAN. The dots denote the inputs/outputs of the hidden layers. The squares indicate the computational operations and the arrows represent the feedforward flow.



(a) Samples in $\mathcal{T}_0$, $\mathcal{T}_1$, and $\mathcal{T}_5$ **(b) $\mathcal{T}_0 \rightarrow \mathcal{T}_1$** **(c) $\mathcal{T}_0 \rightarrow \mathcal{T}_5$**

Figure 2: T-SNE visualization of three tasks, and two different updating trends of weight factors.

Third, the weight factors of deepest layers are likely to stay small. The reason is that once a hidden layer's output becomes very close to the minimizer of the overall loss, the remaining layers that are deeper than this particular layer cannot suggest better prediction results. Since those deepest layers have accumulated substantial loss over T iterations, and they cannot yield smaller loss than that optimal layer does, their weight factors would thus stay small.

Overall, when EDAN is being training, the weight factors of shallow layers first sharply increase. Then, deep layers start to take over and their weights gradually surpass those of shallow layers at certain point. The weights of deepest layers will increase the latest yet remain small. After convergence, if the weight factors of the shallower layers are larger than those of deeper layers, then learning a shallow representation can satisfy Eq. (2), indicating that the candidate task is indeed close to the knowledge base. Otherwise, if the weight factors of deep layers are larger, a complex mapping function is needed to learn more representative latent features (which may lead to negative knowledge transfer), suggesting that the candidate task is quite faraway.

Intuition Verification. A simple example (reduced from D6 in Section 6.1) is given in Figure 2 to illustrate the above intuitions. Figure 2a visualizes the samples of three tasks, one labeled task $\mathcal{T}_0$ and two unlabeled tasks $\mathcal{T}_1$ and $\mathcal{T}_5$, via T-SNE embedding [35]. It is obvious that $\mathcal{T}_0$ is close to $\mathcal{T}_1$ but quite disparate from $\mathcal{T}_5$. Figures 2b and 2c illustrate the updating trends of the weight factors when EDAN learns a pair of two similar tasks (*i.e.*, $\mathcal{T}_0$ and $\mathcal{T}_1$) and that of two disparate tasks (*i.e.*, $\mathcal{T}_0$ and $\mathcal{T}_5$), respectively. We observe that, in both figures the weight factors of shallow layers sharply increase at initial iterations. After that, in Figure 2b, shallow layers converge with large weights, indicating that shallow representations are sufficient to align two close tasks. In contrast, in Figure 2c, deep layers dominate, where only a complex feature alignment function can extract the commonality between two disparate tasks. These findings coincide our intuition, justifying that the weight factors of the learned EDAN can quantify the complexity level of the feature alignment function needed for extracting the common space between two tasks. As such, those weight factors soon deliver us a new metric for the domain-wise distance calibration.

The Metric. For more effectively curriculum design, we reduce the weight factors of an EDAN to a single value, as the weight factors of the hidden layers are vectors and meaningfully ordering them is not easy. We use the idea of *weighted entropy* [19] for such reduction. Specifically, we define $Q = -\sum_{l=1}^{L} l \cdot \alpha_{(l)} \log \alpha_{(l)}$ as the domain-wise distance metric. Conceptually, if Q is small, then shallow layers dominate and play more important roles in final predictions while the deep ones are trivial. If Q is large, then either the weights of all layers are uniformly distributed or those of deep layers are large, both of which mean that the trained network is indeed deep. With these Qs, our approach organizes the candidate tasks into a curriculum and chooses the task that yields the minimal Q to learn in each time step.

4.3 Knowledge Base Augmentation

After choosing a candidate task from the buffer, we use the classifiers of EDAN to predict its instance labels. The predicted instances need to be stored into the knowledge base, such that the new knowledge is continuously accumulated. As such, those tasks, which are quite disparate from the initially labeled $\mathcal{T}_0$, become learnable, since an augmented knowledge base is more likely to share commonality with those disparate tasks. The common space can then be easily extracted without needing an overly complex feature alignment function, circumventing the negative knowledge transferring issue.

However, simply throwing all predicted instances into the knowledge base may not work and is limited by two aspects. On the one hand, the knowledge base itself, storing all data as more tasks being learned, would soon grow to an unmanageably large size. This would lead to memory overhead in a lifelong learning setting, as when the task sequence would stop inputting remains unknown.

On the other hand, the instances per task are predicted with different confidence levels. Consider, for example, that an adversary arranges the tasks in a reversed sequence, in the sense that those tasks being disparate to the original labeled $\mathcal{T}_0$ arrive before the more similar tasks. Under such circumstances, all candidate tasks in the buffer may be not sufficiently close to $\mathcal{T}_0$ in the initial time steps. As a result, the instances of the selected task are predicted with uncertainty. Integrating those instances being predicted with low confidence into the knowledge base is likely to introduce *noises*, which will hurt the learning performance of the subsequent tasks.

To restrict the knowledge base under a manageable size and to eliminate the noises, we choose to integrate those instances that are most confidently predicted. We employ the margin-maximum principle [57] to find out such instances, defined as

$$\max_{\mathrm{x}\sim\mathcal{T}_i} \sum_{l=1}^{L} \alpha_{(l)} \cdot \hat{y}_{(l)} \cdot \frac{\theta_{y\,(l)}^{\top} \mathrm{h}_{(l)}}{\|\theta_{y\,(l)}\|_2}, \tag{6}$$

where $\mathcal{T}_i$ is the learned task at the current time step and $\mathrm{h}_{(l)}$ denotes the feature representation of x output from the l-th hidden layer. From Eq. (6), we observe that the margin for the input x is a weighted sum of the sub-margins, each of which is independently calculated based on the output of a specific hidden layer.

From a geometric viewpoint, the larger the margin is, the more faraway from the decision hyperplane this instance locates. In Eq. (6), $\theta_{y\,(l)}$ represents a vector being orthogonal to the decision hyperplane. Choosing the instances with the largest margin is therefore equivalent to selecting those that are predicted with the least uncertainty.

The process of EDAN training, curriculum design, and knowledge base augmentation continues until no more future task arrives. Through this way, all tasks are learned in a desired ordering, ending up with decent overall learning performance.

5 THEORETICAL ANALYSIS

In this section, we borrow the *regret* from online learning [9] and the *generalization risk* from multi-source domain adaption [6, 14] to analyze the theoretical properties of our approach. The proofs are deferred to the supplemental material. By the analyses, we aim to answer two research questions.

First, we observe that EDAN makes predictions by ensembling the solutions from all layers. Suppose there exists an oracle knowing the optimal depth of training a domain adversarial network for each arriving task in a foresight. The learning performance of such an oracle-supervised domain adversarial network naturally represents the global optimum, necessitating to answer the first question.

Q1. *How does the learning performance of EDAN compare to such an oracle-supervised domain adversarial network?*

Theorem 5.1. *Denoted by $\mathcal{L}_{\mathrm{EDAN}} = \mathbb{E}_{(\mathrm{x},y)\in\mathcal{T}_0}[\ell(y,\hat{y})]$ the empirical risk of EDAN. Suppose the $\star$-th hidden layer is a hindsight optimum, yielding the minimal empirical risk defined as $\mathcal{L}_{\mathrm{ORC}} = \mathbb{E}_{(\mathrm{x},y)\in\mathcal{T}_0}[\ell(y,\hat{y}_{(\star)})]$. With parameter $\tau = 8\sqrt{1/\ln T}$, we have*

$$\mathcal{L}_{\mathrm{EDAN}} < \mathcal{L}_{\mathrm{ORC}} + \frac{\ln L}{T(1-e^{-\tau})}, \tag{7}$$

where T denotes the number of training iterations.

This theorem answers **Q1**, as it states that the empirical risk $\mathcal{L}_{\mathrm{EDAN}}$ is comparable to $\mathcal{L}_{\mathrm{ORC}}$ and is bounded by a small scalar. Note, in practice we do not have the oracle, so the layer yielding the optimal prediction cannot only be obtained in a foresight. Therefore, Theorem 5.1 gives an upper bound of the empirical risk of EDAN. In effect, EDAN enjoys a lower empirical risk than a domain adversarial network with depth chosen in an ad-hoc way.

Second, in addition to EDAN, the other important building block of our approach is the curriculum learning strategy, where i) a knowledge base $\mathrm{R}^{(i)}$ accumulates knowledge from the tasks learned in a desired ordering and ii) a lifelong learner being trained on the $\mathrm{R}^{(i)}$ predicts the next chosen task.

Specifically, suppose $i-1$ tasks have been learned in such a curriculum that $d_{\mathcal{H}}(\mathcal{T}_0,\mathcal{T}_1) \le d_{\mathcal{H}}(\mathcal{T}_0,\mathcal{T}_2) \le \ldots \le d_{\mathcal{H}}(\mathcal{T}_0,\mathcal{T}_{i-1})$, where $d_{\mathcal{H}}(\cdot,\cdot)$ is $\mathcal{H}$-divergence representing the distance between two tasks over the hypothesis space $\mathcal{H}$. By our approach, the knowledge base $\mathrm{R}^{(i)}$ embodies the instances from those $i-1$ tasks $\{\mathcal{T}_1,\ldots,\mathcal{T}_{i-1}\}$. Let $\hat{h} \in \mathcal{H}$ denote the hypothesis learned from $\mathrm{R}^{(i)}$.

Now, a new task $\mathcal{T}_i$ arrives, where the groundtruth hypothesis h^* underlies. Note, no such h^* can be obtained in practice, as $\mathcal{T}_i$ has no label. Comparing the empirical risks of $\hat{h}$ and h^* suffered on $\mathcal{T}_i$ allows to answer:

Q2. *Does lifelong learning with curricula lead to prediction performance improvement?*

Theorem 5.2. *Denoted by $\epsilon_{\mathcal{T}_i}(\hat{h})$ and $\epsilon_{\mathcal{T}_i}(h^*)$ the empirical risks suffered by using $\hat{h}$ and h^* to predict data in $\mathcal{T}_i$, respectively. We have*

$$\epsilon_{\mathcal{T}_i}(\hat{h}) \le \epsilon_{\mathcal{T}_i}(h^*) + d_{\mathcal{H}}(\mathrm{R}^{(i)},\mathcal{T}_i) + K, \tag{8}$$

where $d_{\mathcal{H}}(\mathrm{R}^{(i)},\mathcal{T}_i) \le d_{\mathcal{H}}(\mathcal{T}_0,\mathcal{T}_i)$ and K is a scalar bounded by $\sqrt{\log(2|\mathrm{R}^{(i)}|)/|\mathrm{R}^{(i)}|}$.

Also, we let $h_{\mathcal{T}_0}$ be the hypothesis learned from $\mathcal{T}_0$ directly, where neither curriculum learning is involved nor knowledge base is constructed.

Proposition 5.3. *Let $\epsilon_{\mathcal{T}_i}(h_{\mathcal{T}_0})$ denote the empirical risk suffered by using $h_{\mathcal{T}_0}$ to predict data in $\mathcal{T}_i$. For any time step $i > 0$, as $|\mathrm{R}^{(i)}| > |\mathcal{T}_0|$, we have $\epsilon_{\mathcal{T}_i}(\hat{h}) < \epsilon_{\mathcal{T}_i}(h_{\mathcal{T}_0})$.*

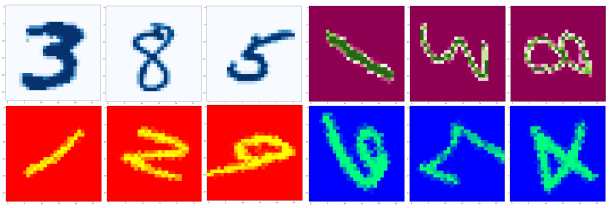


Figure 3: Illustration of four tasks in MNIST-Rainbow (D1).

Task	Review
Movies	"Cheap CG. The fight is on a not-so-epic, toy-like representation of Moon." (−)
Toys	"I prefer wood or cloth toy to plastic so I love that these be soft and crinkle." (+)
Headphones	"There be a sort of echo in the cup that sound like one of those kid microphone toy." (−)

Figure 4: Demonstration of three tasks and corresponding reviews in Amazon Product Review (D2).

Theorem 5.2 and Proposition 5.3 together answer **Q2** from two aspects. i) Theorem 5.2 states that using a hypothesis trained from $\mathrm{R}^{(i)}$ for predicting $\mathcal{T}_i$ yields comparable performance with the groundtruth hypothesis underlying $\mathcal{T}_i$. As more tasks are handled, both the $\mathcal{H}$-divergence term and the scalar K get smaller, tightening up this performance bound. ii) Proposition 5.3 shows that predicting $\mathcal{T}_i$ with a hypothesis trained on $\mathrm{R}^{(i)}$ enjoys a lower empirical risk than that trained on $\mathcal{T}_0$. This finding coincides the intuition that, as $\mathrm{R}^{(i)}$ embodies additional instance (representing knowledge) from the learned tasks $\{\mathcal{T}_1, \ldots, \mathcal{T}_{i-1}\}$, it is more likely to be close to $\mathcal{T}_i$, such that the common space between $\mathrm{R}^{(i)}$ and $\mathcal{T}_i$ can be easily extracted. The negative knowledge transferring issue can be circumvented, improving the overall learning performance.

6 EVALUATION

In this section, we present empirical evidences to substantiate the viability and effectiveness of our proposed approach. Specifically, Section 6.1 introduces the studied datasets. Section 6.2 elaborates the experiment setup. Experimental results are given in Section 6.3.

6.1 Datasets

We benchmark the experiments on six datasets, among which five datasets are widely used in the lifelong learning literature and one dataset is synthesized by ourselves. Notably, the datasets are in different modalities including images (*i.e.*, D1), natural languages (*i.e.*, D2), sensory data (*i.e.*, D4), and structured data (*i.e.*, D3, D5, and D6), validating the promise of our approach in generalizing to a wide range of web application fields.

MNIST-Rainbow (D1) is created by [15], comprising 56 tasks. Each task has 900 images with 28×28 pixels, transformed from the original hand-written digits with various color-mapping, shearing, rescaling, and rotating. Figure 3 illustrates four tasks in this dataset to show their disparateness.

Amazon Product Review (D2) is introduced by [13], which includes reviews crawled from 20 types of diverse products in Amazon (*i.e.*, 20 tasks). Each task has 2000 reviews (instances), and the learning tasks are to classify those reviews into positive (rating > 3) or negative (rating < 3) sentiments. Figure 4 demonstrates three tasks in which it is worth to note that the same word "toy" conveys disparate semantic meanings across tasks.

Linux Kernel Codebase (D3) is collected by [54], which contains 21,193 source code paths from 10 projects written in the C language, including Linux, libc, *etc.*. Since each project was built by a separate group, it is reasonable to consider paths that are from a single project as data instances that form an individual task. Each path is encoded by 13 features. The goal is to predict whether each path is an error path (or not).

Land Mine Detection (D4) includes 14,820 data instances associated with 9 features, which were captured by radars located in 29 different geographical regions [56]. Each data instance refers to an area in a specific region, and the goal is to detect whether a land mine is present in an area or not. We treat data instances collected from each single region as a different task. This dataset and the following were also used in [43] for evaluation.

London School Data (D5) consists of examination performances (pass or fail) from 15,362 students in 139 schools in London [30]. Scores for students from an individual school are treated as data instances in a single disparate task. Each student is described by 27 features, and the goal is to predict the examination results (pass or fail) for all students.

Synthetic Classification Tasks (D6) contains 11 binary tasks with 10 features and 1000 data instances per task. One task was randomly selected as $\mathcal{T}_0$, and the remaining unlabeled tasks ($\mathcal{T}_1$ through $\mathcal{T}_{10}$) are deemed as the input task sequence. To simulate disparate data distributions, we mapped the features in the unlabeled tasks, following the idea in [52]. For $\mathcal{T}_i$, its i out of 10 features are mapped with random Gaussian matrices. As such, we have a-prior knowledge that $\mathcal{T}_1$ is the closest to $\mathcal{T}_0$ (as only one feature in $\mathcal{T}_1$ is mapped), followed by $\mathcal{T}_2$, and $\mathcal{T}_{10}$ is the most faraway task.

Notably, in each of the first five datasets (*i.e.*, D1 – D5), the data distributions of tasks are naturally disparate from each other. We randomly pick one task to label as the initial task and keep other tasks remain unlabeled as the input task sequence. However, in each dataset, we do not know which tasks are more similar to the initial task in advance. To validate whether the tasks are indeed ordered in a desired curriculum, D6 is synthesized in which we know in a priori that the distances from the input tasks to $\mathcal{T}_0$ monotonically increase in a sequence of $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_{10}$.

6.2 Experiment Setup

6.2.1 Compared Methods. We evaluate our approach against five state-of-the-art methods. In the below, we describe their high-level ideas and discuss why we choose them as baselines.

- **UDA** [17] and **DAN** [33], both of which are unsupervised domain adaptation methods, learn the latent common space between two tasks with separately devised domain adversarial networks. Their difference is that UDA relies on the output from the last hidden layer in making predictions while DAN considers outputs from all hidden layers.

- **GeppNet** [18] builds a dual-memory network to deal with multiple tasks in an incremental fashion. It stores data of each learned task and regularly replays old data interleaved with new data to mitigate catastrophic forgetting.
- **EWC** [29] preserves learned knowledge by penalizing parameter changes, and allows new tasks to update only those parameters that are less important to old tasks.
- **GEM** [34] stores subsets of samples from previous tasks as episodic memories, which are used to regularize the gradient when processing a new task – the gradient is projected onto a new direction that minimizes the loss suffered on new task and does not increase the losses on episodic memories.

The reasons why we choose those competitors are as follows. On the one hand, UDA and DAN share the same idea of employing domain adversarial networks to predict the unlabeled tasks, However, UDA and DAN focused on a single-source-single-target setting, where the labeling information is propagated from the initially labeled task to each and every unlabeled tasks only. They did not accumulate knowledge from the learned unlabeled tasks and, as a result, the negative knowledge transfer is likely to be incurred if an arriving task is quite faraway. Comparing with UDA and DAN, our approach accumulates knowledge from the learned tasks, enjoying potential improvement of learning performance.

On the other hand, GeppNet, EWC, and GEM represent the state-of-the-art lifelong learning (continual learning) methods, which use diverse strategies to retain knowledge from the learned tasks. EWC belongs to the model-based lifelong learning family which focus on regularizing the model parameters but do not store historical data, while GeppNet and GEM are the rehearsal-based lifelong learners in which the historical data are stored in an external memory. It is thus beneficial to compare with such diverse strategies, so as to investigate which is the best practice for knowledge retention in our setting. Note that all the three methods are supervised, requiring labeled data in all tasks, while our approach entails one labeled task only. A comparison with them reveals whether our unsupervised lifelong learner can attain a comparable performance with those supervised counterparts, directly testing the tightness of our Theorem 5.2.

6.2.2 Parameter Settings. For our approach, we initialize the over-complete EDAN with 10 hidden layers and 100 units per layer. `ReLU` is employed for the common features extraction and `softmax` and `sigmoid` are used by the classifiers and task discriminators for predicting the labels and task memberships, respectively. ADAM optimizer with a step size of 10^{-5} decaying in 5e−4 and a mini-batch of 64 is used for training EDAN. The weight factor $\alpha_{(l)}$ is initialized as 0.1 for each layer, and is updated with Eq. (5). The discount rate τ is initialized 0.85 and decays with the rule suggested by Theorem 5.1. We configure the baselines as follows. For UDA and DAN, three different network architectures are built with 3, 6, and 10 hidden layers. For GeppNet, EWC, and GEM, three ratios (10%, 30%, and 50%) of labeled data are given in all tasks. Other parameters are set as suggested in the respective literatures.

6.2.3 Evaluation Protocol. For GeppNet, EWC, and GEM where labels are required in all tasks, 20% instances in each task (except the initially labeled task) are held-out as a test set. The tasks are input in an incremental manner. After all tasks have been learned, the labels of the test set are revealed, over which the the accuracy is calculated. We shuffle the input task sequence 10 times to repeat the experiments and report the average results.

For UDA, DAN, and our approach, the algorithm is given a buffer of unlabeled tasks at each iteration, from which one task is chosen to learn. For UDA and DAN, the choice is random; For our approach, the choice is based on the curriculum. UDA and DAN starts the next iteration directly without knowledge retention, while our approach starts the next iteration after augmenting the knowledge base. Once all unlabeled tasks are learned, the true labels are revealed, and the accuracy is calculated across all predicted data.

In addition to the accuracy, which evaluate the algorithm performance in an end-to-end fashion, we introduce a new metric, termed forward transfer error (FTE), defined as:

$$\text{FTE} = \frac{1}{|\mathcal{T}_F|} \sum_{i=1}^{N} \sum_{j=1}^{|\mathcal{T}_F|} \left[y_j \neq \hat{h}_i(\phi_i(\mathrm{x}_j)) \right], \tag{9}$$

where $\mathcal{T}_F$ denote the task that is most faraway from the original labeled task. Denoted by $\hat{h}_i$ and ϕ_i the hypothesis and the feature alignment function learned at the i-th time step, respectively. The intuition behind FTE is that, after learning each candidate task, we enforce the learner to predict the most disparate task. If the learner gradually grows its knowledge to become more competent in dealing with the most disparate task, FTE should get lower over time. Otherwise, FTE fluctuates arbitrarily, meaning that the learner does not retain any useful knowledge from the past learning iterations.

Moreover, the buffer size of our approach may yield an accuracy-efficiency tradeoff. Conceptually, among a larger buffer, it is more likely to choose a task that the learner can predict with the highest accuracy, where learning with curriculum is more helpful. However, a larger buffer will consume longer time to order the candidate tasks within it. To calibrate this tradeoff, we apply two buffer sizes, dividing our approach into two variants. One is termed ULLC-A(ccuracy), the other is termed ULLC-S(peed). In ULLC-A, we use large buffers, with the buffer size for D2, D3, D4, and D6 being 20 and for D1 and D5 being 50. In ULLC-S, the buffer size is small, which is fixed as 5 for all datasets.

6.3 Results

We present the experimental results in this section, aiming to answer four research questions (**Q3** – **Q6**) as follows.

Q3. *How does our approach compare to the state-of-the-arts?*

To answer this question, we present the classification results of the two variants of our approach, namely, ULLC-A and ULLC-S, along with those of the competitors in Table 1. All the results are in the format of mean accuracy ± standard deviation, obtained from running each experiment 10 times. To show the statistical significance, we carry out a paired t-test on the results. If a result of our approach outperforms the compared methods with hypothesis supported at 95% significance level, we count a "win". If our result outperforms but does not surpass the 95% significance level, we count a "tie". Otherwise, we count a "loss". The win/tie/loss counts are summarized at the two bottom rows in Table 1.

From the table, we make three observations. *First*, ULLC-A significantly outperforms UDA and DAN in all settings with a 23.48%

Table 1: Experimental results in format of mean accuracy ± standard deviation (%). In the parentheses, "L" denotes the maximal network depth while "%" indicates the ratio of labeled instances. The results are from 10 experiment repeats. ULLC-A and ULLC-S are our approaches. The win/tie/loss counts are summarized in the last two lines.

Label ?	Method	D1	D2	D3	D4	D5	D6
Labels In All Tasks	GeppNet (10%)	59.0 ± 6.5	61.4 ± 4.4	65.5 ± 6.3	67.7 ± 5.8	66.4 ± 5.4	68.6 ± 9.2
	GeppNet (30%)	65.7 ± 6.1	68.0 ± 3.2	72.0 ± 4.1	76.8 ± 4.9	83.7 ± 4.9	74.6 ± 5.6
	GeppNet (50%)	79.8 ± 5.2	74.3 ± 3.1	78.3 ± 5.7	82.5 ± 4.2	86.3 ± 4.6	82.1 ± 6.8
	EWC (10%)	67.5 ± 1.6	61.8 ± 1.4	71.7 ± 1.3	73.6 ± 0.7	69.4 ± 0.9	77.3 ± 1.3
	EWC (30%)	76.0 ± 1.9	69.3 ± 0.8	82.1 ± 1.3	84.5 ± 2.2	80.0 ± 0.9	86.1 ± 1.5
	EWC (50%)	86.2 ± 1.2	78.2 ± 1.6	87.3 ± 1.1	89.2 ± 1.9	84.6 ± 1.1	92.9 ± 1.2
	GEM (10%)	76.3 ± 2.1	62.5 ± 1.0	74.5 ± 2.6	71.0 ± 3.1	72.3 ± 1.6	75.2 ± 2.7
	GEM (30%)	82.6 ± 1.8	72.4 ± 1.1	86.7 ± 2.7	84.2 ± 2.0	85.7 ± 2.1	87.3 ± 1.9
	GEM (50%)	90.3 ± 0.5	84.7 ± 0.9	90.9 ± 1.2	86.0 ± 2.4	89.6 ± 1.6	94.8 ± 1.8
No Label in $\mathcal{T}_1, \ldots, \mathcal{T}_N$	UDA ($L = 3$)	69.3 ± 1.8	63.2 ± 2.0	58.1 ± 2.3	63.4 ± 1.7	61.4 ± 1.4	65.6 ± 2.2
	UDA ($L = 6$)	79.4 ± 2.2	65.7 ± 1.6	57.4 ± 1.9	71.4 ± 0.9	65.1 ± 1.2	64.9 ± 1.9
	UDA ($L = 10$)	76.8 ± 0.5	65.3 ± 0.8	52.6 ± 1.7	51.0 ± 1.5	68.2 ± 1.5	50.3 ± 1.1
	DAN ($L = 3$)	74.2 ± 2.5	68.4 ± 0.5	61.8 ± 2.3	72.6 ± 1.7	69.8 ± 1.7	70.4 ± 2.2
	DAN ($L = 6$)	72.7 ± 0.8	66.9 ± 0.2	63.2 ± 1.9	68.8 ± 0.9	71.5 ± 1.9	67.5 ± 1.9
	DAN ($L = 10$)	74.6 ± 1.1	70.1 ± 0.2	51.7 ± 1.7	50.0 ± 1.5	52.0 ± 1.2	50.1 ± 1.1
	(Ours.) ULLC-A	85.4 ± 1.1	75.8 ± 0.4	86.5 ± 0.7	82.4 ± 0.4	83.3 ± 1.3	89.4 ± 0.8
	(Ours.) ULLC-S	80.7 ± 2.9	71.3 ± 1.5	78.6 ± 5.4	75.7 ± 4.1	75.4 ± 2.2	81.3 ± 8.6
W/T/L	ULLC-A(ccuracy) *	10 / 3 / 2	11 / 2 / 2	11 / 1 / 3	10 / 0 / 5	9 / 1 / 5	10 / 3 / 2
	ULLC-S(peed)	6 / 6 / 3	8 / 3 / 4	7 / 4 / 4	5 / 4 / 6	7 / 2 / 6	9 / 1 / 5

* ULLC-A wins unsupervised methods in all settings and only loses when the counterpart is a supervised method with a 50% (and sometimes 30%) of labeling.

increased accuracy in average. This reveals that storing instances from the learned tasks helps grow the knowledge of the lifelong learner, such that the overall prediction performance is improved.

Second, for UDA and DAN, the optimal network depth that yields the best performance varies across different datasets. This necessitates a machinery that adaptively tunes the neural architecture in accordance with the data complexity, instead of setting the network depth in a priori. Our devised EDAN provides such a machinery and achieves decent empirical performance. This finding coincides with our theoretical analysis in Theorem 5.1, showing that the adaptive nature of our EDAN could achieve comparable (better, empirically) results than those domain adversarial networks whose depths are set in an ad-hoc manner.

Third, ULLC-A wins GeppNet in 13 out of 18 settings across all datasets, increasing the classification accuracy by 13.28%. Compared with EWC and GEM, overall, ULLC-A wins when 10% labels are given, ties when 30% labels are given, and loses when 50% labels are available. These findings indicate that ULLC-A is comparable to the

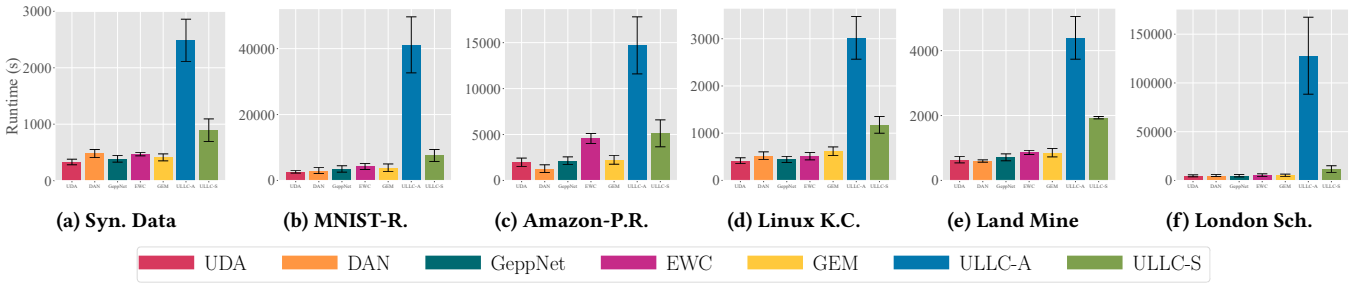


Figure 5: Runtime performance of our ULLC-A and ULLC-S approaches with their competitors in second(s). The colored bars correspond to UDA, DAN, GeppNet, EWC, GEM, ULLC-A, and ULLC-S, in sequence (as shown in the legend at the bottom panel). The variances are indicated in lined intervals, attached on top of the bars.

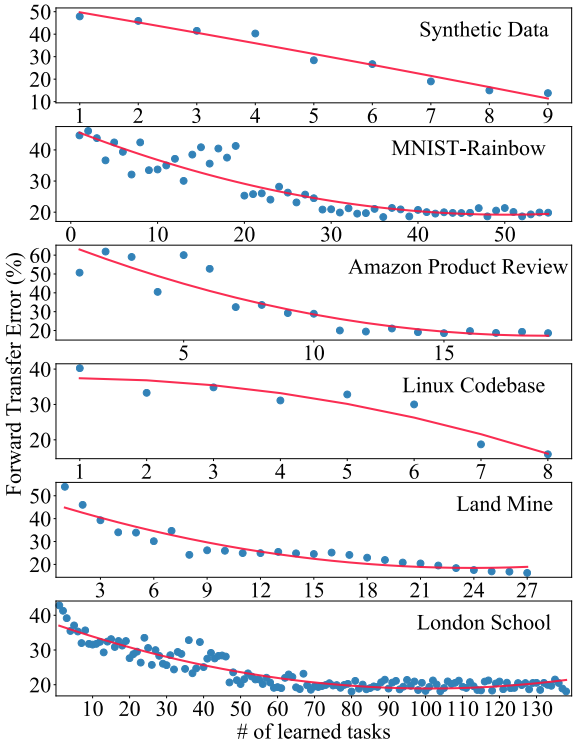


Figure 6: The trends of FTE in predicting the most disparate task with respect to the number of learned tasks. Red lines are the best fitting exponential curves.

state-of-the-art lifelong learning methods that are supervised and require a complete labeling information in all tasks. The fact that ULLC-A can perform accurate predictions with no label needed in any future tasks promises the optimism of applying ULLC-A to real applications where the labeling budget is very limited.

Q4. *What is the tradeoff between accuracy and efficiency?*

Intuitively, our two algorithms both consume more computation time than the compared methods, where the extra runtimes mainly amount for ordering the candidate tasks into the curricula. To gauge this runtime overhead, we have compared the runtime performance of our approaches to the compared methods across all datasets, as illustrated in Figure 5.

We first observe that our ULLC-A algorithm suffers from around 14× slowdown in D1, 6× slowdown in D2, D3, D4, and D6, and 35× slowdown in D5, when compared with its counterparts. The main overhead lies in re-ordering a large number of candidate tasks at each time step, as the buffer size applied in ULLC-A is quite large.

In case efficiency becomes critical, our ULLC-S algorithm, using a small and fixed buffer size, can be readily applied. As shown in Figure 5, we observe that the slowdowns of ULLC-S in average are 3× in D1, D2, D3, D4, and D6 and 8× in D5, which is much improved than ULLC-A. Although a small buffer may lead to inferior performance, from Table 1, we observe that such an accuracy tradeoff is not significant in practice. Specifically, compared with ULLC-A, ULLC-S sacrifices a 7.6% accuracy in average to gain the computation efficiency. It is worth to point out that ULLC-S still outperforms GeppNet (10%), EWC (10%), and GEM (10%) in all datasets, outperforms GeppNet (30%) with two exceptions only, and outperforms EWC (30%) in two datasets.

Overall, the accuracy-efficiency tradeoff exists in our approach. We have proposed two variants, namely, ULLC-A and ULLC-S, as a countermeasure, where ULLC-A yields better classification accuracy by paying off a longer runtime and ULLC-S attains a slightly inferior accuracy but enjoys a much improved computation efficiency. One can easily configure which variants to be applied to suit specific application requirements, either focusing more on the accuracy or on the efficiency.

It is worth to note that, our approach operates in an *unsupervised* fashion, in the sense that it does not require a continuous human supervision to provide any labels for the future tasks. We believe such an accuracy-efficiency tradeoff pays off since, in practice, even a proportional labeling effort is onerous. For example, in our experiments where the datasets contain from 10,000 to 21,193 instances, even 10% labels would entail tedious and costly human efforts. Our approach provides an apparatus to avoid the human labeling overhead.

Q5. *How do the curriculum design and knowledge retention improve the learning accuracy?*

We study this question from two perspectives. *First*, we illustrate the trends of forward transfer error (FTE), an evolution trends of classification error rate at the most disparate task as more tasks are learned, as shown in Figure 6. Specifically, for D6, by construction, $\mathcal{T}_{10}$ is the most disparate task. For other datasets, as no such information is available, we treat the task that is lastly learned in the curriculum as the most disparate task.

From the figure, we observe that our approach can gradually make more accurate predictions on the most disparate task as more candidate tasks have been learned. This phenomenon demonstrates that our lifelong learner indeed becomes competent for predicting those tasks that it was originally unfamiliar with (*i.e.*, those tasks being disparate from the knowledge base at initial time steps), by growing its knowledge gradually with a curriculum.

Second, from Table 1, we observe that, GeppNet suffers from high variance with few exceptions. The reason is that, GeppNet imposes weak constraint when updating network parameters, thereby being brittle to the input task sequence. If a disparate task is handled early by GeppNet, then the old knowledge is overwritten, deteriorating the accuracy of GeppNet in predicting the pervious tasks. Our approach overall enjoys both a higher accuracy and a lower variance over GeppNet, demonstrating that learning tasks in a meaningful curriculum can improve the overall prediction performance.

Q6. *Is the weighted entropy a good heuristic for ordering candidate tasks in curriculum?*

We leverage D6 to answer this question at two levels. At the coarse level, we know in a priori that the unlabeled tasks $\mathcal{T}_1$ through $\mathcal{T}_{10}$ are increasingly further away from the labeled $\mathcal{T}_0$ in D6. Thus, if the weighted entropy can precisely order tasks, then the curriculum learned by our approach should coincide with this a prior task ordering in D6. At the fine level, we enforce the learner to use the knowledge base formed at each time step to predict the most disparate task $\mathcal{T}_{10}$, which is similar to the setting in Q5. During this

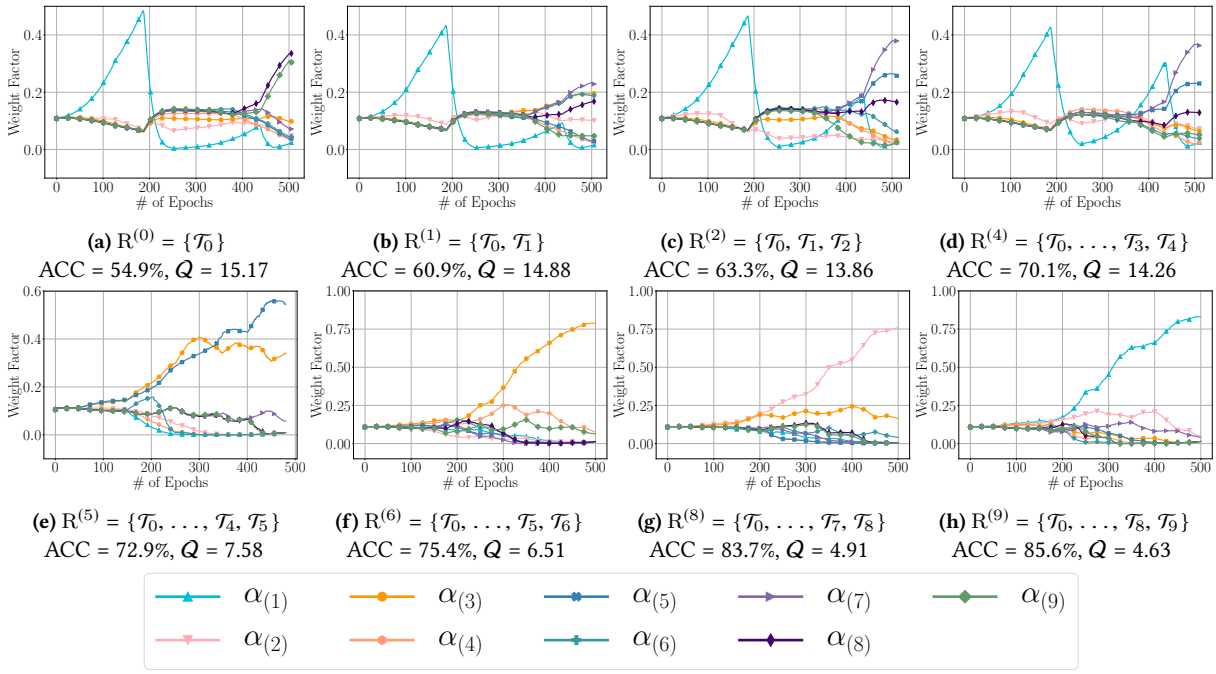


Figure 7: Illustration of the task learning order (curriculum) in D6. The figures plot the trends of the weight factors during training EDAN, where $\alpha_{(1)}$ and $\alpha_{(9)}$ correspond to the shallowest and the deepest layers, respectively. The captions indicate the tasks learned at each time step and calculate the weighted entropy Q based on the weight factors after convergence and the classification accuracy by applying the learner to $\mathcal{T}_{10}$.

process, we plot the weight factors of the learned EDAN and record the corresponding weighted entropies (*i.e.*, the Qs) and prediction accuracies (*i.e.*, the *ACC*s) in Figure 7. If the weighted entropy is a good heuristic, it should precisely characterizes the changes of the weight factors along the time horizon and builds a direct link between the weight factors and the accuracies.

The curriculum that the ULLC-A algorithm learned for 8 time steps is illustrated Figure 7 (We omitted the figures for two steps to save space). Our observations are as follows. *First*, the captions of the sub-figures 7a – 7h describe the task that is chosen and is merged into the knowledge base at each time step, representing the curriculum learned by our approach. From the captions, we observe that the tasks are chose in the ordering of $\mathcal{T}_1$, followed by $\mathcal{T}_2$, and all the way to $\mathcal{T}_9$. This ordering coincides with the a prior task ordering in D6, verifying that a desired curriculum is obtained with the help of weight entropy.

Second, the weight entropy is strongly correlated with the prediction accuracies and the weighted factors of trained EDAN. Specifically, in sub-figures 7a – 7d, as the knowledge base remains far-away from $\mathcal{T}_{10}$, the deeper layers in EDAN dominate, which is precisely characterized by the higher-valued weighted entropies. In sub-figures 7e – 7h, once the knowledge base approaches $\mathcal{T}_{10}$, shallow layers start to take over, which is reflected by the weighted entropies with decreasing values. Meanwhile, the pattern is clear that, as the value of weighted entropy goes down, the prediction accuracy increases. This finding confirms that the weighted entropy is a good heuristic to reduce the weight factors (vectors) into a single value to ease task ordering, as we desired.

7 CONCLUSION

This paper proposed a novel lifelong learning paradigm, named unsupervised lifelong learning with curricula (ULLC), which enables a learning system to continuously learn from a sequence of future tasks that are quite disparate from the initial task it was trained on. As labeling is a costly, tedious, and error-prone in practice, providing constant labeling efforts for all future tasks is close to impossible. Our paradigm that requires no labels from any future task is thus particularly promising. The main challenge in our paradigm lies in the occurrence of negative knowledge transfer, as it is fundamentally difficult to identify which pieces of previously learned knowledge are helpful or detrimental for learning a future task without the presence of task labels. Our key idea to overcome this challenge is curriculum learning. That is, instead of selecting the knowledge pieces to best match a given task, our learner actively selects the next task to learn, based on its distance to the knowledge base, which is formed by retaining and accumulating data instances that are predicted with high confidence from earlier tasks. A theoretical analysis substantiated that our curriculum learning idea provably leads to performance improvements over other lifelong learners that deal with tasks in an arbitrary order. An empirical study with various real-world datasets shows that our approach achieves comparable performance to its supervised learning counterparts, confirming the viability of our approach.

REFERENCES

[1] Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya, and Gerhard Weikum. 2018. Never-ending learning for open-domain question answering over knowledge bases. In *WWW*. 1053–1062.

[2] Arvind Agarwal, Samuel Gerber, and Hal Daume. 2010. Learning multiple tasks using manifold regularization. In *NIPS*. 46–54.

[3] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. 2017. Expert gate: Lifelong learning with a network of experts. In *CVPR*. 3366–3375.

[4] Maria-Florina Balcan and Avrim Blum. 2005. A PAC-style model for learning from labeled and unlabeled data. In *COLT*. Springer, 111–126.

[5] María Teresa Ballestar, Pilar Grau-Carles, and Jorge Sainz. 2019. Predicting customer quality in e-commerce social networks: a machine learning approach. *Review of Managerial Science* 13, 3 (2019), 589–603.

[6] John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman. 2008. Learning bounds for domain adaptation. In *NIPS*. 129–136.

[7] Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT*. 92–100.

[8] Zhangjie Cao, Kaichao You, Mingsheng Long, Jianmin Wang, and Qiang Yang. 2019. Learning to transfer examples for partial domain adaptation. In *CVPR*. 2985–2994.

[9] Nicolo Cesa-Bianchi and Gabor Lugosi. 2006. *Prediction, learning, and games*. Cambridge university press.

[10] Shih-Fu Chang, Wei-Ying Ma, and Arnold Smeulders. 2007. Recent advances and challenges of semantic image/video search. In *ICASSP*, Vol. 4. IEEE, IV–1205.

[11] Olivier Chapelle and Ya Zhang. 2009. A dynamic bayesian network click model for web search ranking. In *WWW*. 1–10.

[12] Hsinchun Chen and Michael Chau. 2004. Web mining: Machine learning for web applications. *Annual review of information science and technology* 38, 1, 289–329.

[13] Zhiyuan Chen, Nianzu Ma, and Bing Liu. 2015. Lifelong learning for sentiment classification. (2015).

[14] Koby Crammer, Michael Kearns, and Jennifer Wortman. 2008. Learning from multiple sources. *Journal of Machine Learning Research* 9, Aug (2008), 1757–1774.

[15] Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. 2019. Online Meta-Learning. In *ICML*. 1920–1930.

[16] Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* 55, 1 (1997), 119–139.

[17] Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *ICML*. 1180–1189.

[18] Alexander Gepperth and Cem Karaoguz. 2016. A bio-inspired incremental learning architecture for applied perceptual problems. *Cognitive Computation* 8, 5 (2016), 924–934.

[19] Silviu Guiaşu. 1971. Weighted entropy. *Reports on Mathematical Physics* 2, 3 (1971), 165–179.

[20] Yunhui Guo, Yandong Li, Liqiang Wang, and Tajana Rosing. 2020. AdaFilter: Adaptive Filter Fine-tuning for Deep Transfer Learning. In *AAAI*.

[21] Lei Han and Yu Zhang. 2016. Multi-stage multi-task learning with reduced rank. In *AAAI*.

[22] Shaobo Han, Xuejun Liao, and Lawrence Carin. 2012. Cross-domain multitask learning with latent probit models. In *ICML*.

[23] Yuan Hao, Yanping Chen, Jesin Zakaria, Bing Hu, Thanawin Rakthanmanon, and Eamonn Keogh. 2013. Towards never-ending learning from time series streams. In *KDD*. 874–882.

[24] Wenpeng Hu, Zhou Lin, Bing Liu, Chongyang Tao, Zhengwei Tao, Jinwen Ma, Dongyan Zhao, and Rui Yan. 2018. Overcoming catastrophic forgetting for continual learning via model adaptation. In *ICLR*.

[25] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. 2016. Deep networks with stochastic depth. In *ECCV*. Springer, 646–661.

[26] Yunhun Jang, Hankook Lee, Sung Ju Hwang, and Jinwoo Shin. 2019. Learning what and where to transfer. In *ICML*.

[27] Lu Jiang. 2016. Web-scale multimedia search for internet video content. In *WWW*. 311–316.

[28] Thorsten Joachims. 2003. Transductive learning via spectral graph partitioning. In *ICML*. 290–297.

[29] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.

[30] Abhishek Kumar and Hal Daume III. 2012. Learning task grouping and overlap in multi-task learning. In *ICML*. 1383–1390.

[31] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. 2017. Fractalnet: Ultra-deep neural networks without residuals. In *ICLR*.

[32] Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence* 40, 12 (2017), 2935–2947.

[33] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. 2015. Learning transferable features with deep adaptation networks. In *ICML*. 97–105.

[34] David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In *NeurIPS*. 6467–6476.

[35] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[36] Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matt Gardner, Bryan Kisiel, et al. 2018. Never-ending learning. *Commun. ACM* 61, 5 (2018), 103–115.

[37] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2009), 1345–1359.

[38] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks* 113 (2019), 54–71.

[39] Shafin Rahman, Salman Khan, and Nick Barnes. 2019. Transductive learning for zero-shot object detection. In *CVPR*. 6082–6091.

[40] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *CVPR*. 2001–2010.

[41] Steffen Rendle and Lars Schmidt-Thieme. 2008. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *ACM Recommender systems*. 251–258.

[42] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. 2019. Experience replay for continual learning. In *NeurIPS*. 348–358.

[43] Paul Ruvolo and Eric Eaton. 2013. Active task selection for lifelong machine learning. In *AAAI*.

[44] Doyen Sahoo, Quang Pham, Jing Lu, and Steven CH Hoi. 2018. Online deep learning: Learning deep neural networks on the fly. In *IJCAI*.

[45] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized experience replay. In *ICLR*.

[46] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. 2017. Incremental learning of object detectors without catastrophic forgetting. In *Proceedings of the IEEE International Conference on Computer Vision*. 3400–3409.

[47] Daniel L Silver, Qiang Yang, and Lianghao Li. 2013. Lifelong machine learning systems: Beyond learning algorithms. In *AAAI*.

[48] Tomer Simon, Avishay Goldberg, Limor Aharonson-Daniel, Dmitry Leykin, and Bruria Adini. 2014. Twitter in the cross fire—the use of social media in the Westgate Mall terror attack in Kenya. *PloS one* 9, 8 (2014), e104136.

[49] Partha Pratim Talukdar and Koby Crammer. 2009. New regularized algorithms for transductive learning. In *ECML-PKDD*. Springer, 442–457.

[50] Ben Tan, Yangqiu Song, Erheng Zhong, and Qiang Yang. 2015. Transitive transfer learning. In *KDD*. 1155–1164.

[51] 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 *ICLR*.

[52] Chang Wang and Sridhar Mahadevan. 2011. Heterogeneous domain adaptation using manifold alignment. In *IJCAI*.

[53] Zirui Wang, Zihang Dai, Barnabás Póczos, and Jaime Carbonell. 2019. Characterizing and avoiding negative transfer. In *CVPR*. 11293–11302.

[54] Baijun Wu, John Peter Campora III, Yi He, Alexander Schlecht, and Sheng Chen. 2019. Generating precise error specifications for C: a zero shot learning approach. In *OOPSLA*. ACM, 160–191.

[55] Yonghui Xu, Sinno Jialin Pan, Hui Xiong, Qingyao Wu, Ronghua Luo, Huaqing Min, and Hengjie Song. 2017. A unified framework for metric transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 29, 6 (2017), 1158–1171.

[56] Ya Xue, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. 2007. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research* 8, Jan (2007), 35–63.

[57] Ziang Yan, Yiwen Guo, and Changshui Zhang. 2019. Adversarial Margin Maximization Networks. *IEEE transactions on pattern analysis and machine intelligence* (2019).

[58] Wei Ying, Yu Zhang, Junzhou Huang, and Qiang Yang. 2018. Transfer learning via learning to transfer. In *ICML*. 5085–5094.

[59] Yizhou Zhang, Guojie Song, Lun Du, Shuwen Yang, and Yilun Jin. 2019. DANE: Domain Adaptive Network Embedding. In *IJCAI*.

[60] Yu Zhang and Dit-Yan Yeung. 2010. Transfer metric learning by learning task relationships. In *KDD*. 1199–1208.

[61] Yu Zhang and Dit-Yan Yeung. 2011. Multi-task learning in heterogeneous feature spaces. In *AAAI*.

[62] Dengyong Zhou and Christopher JC Burges. 2007. Spectral clustering and transductive learning with multiple views. In *ICML*. 1159–1166.