# Online Task-free Continual Learning with Dynamic Sparse Distributed Memory

Julien Pourcel[1], Ngoc-Son Vu[1], and Robert M. French[2]

[1] ETIS - CYU, ENSEA, CNRS, France
[2] LEAD - UBFC, CNRS, France

**Abstract.** This paper addresses the very challenging problem of online task-free continual learning in which a sequence of new tasks is learned from non-stationary data using each sample only once for training and without knowledge of task boundaries. We propose in this paper an efficient semi-distributed associative memory algorithm called Dynamic Sparse Distributed Memory (DSDM) where learning and evaluating can be carried out at any point of time. DSDM evolves dynamically and continually modeling the distribution of any non-stationary data stream. DSDM relies on locally distributed, but only partially overlapping clusters of representations to effectively eliminate catastrophic forgetting, while at the same time, maintaining the generalization capacities of distributed networks. In addition, a local density-based pruning technique is used to control the network's memory footprint. DSDM significantly outperforms state-of-the-art continual learning methods on different image classification baselines, even in a low data regime. Code is publicly available: https://github.com/Julien-pour/Dynamic-Sparse-Distributed-Memory

## 1 Introduction

Although having obtained impressive performance on many different individual tasks, current static deep neural networks (DNNs) must be trained on batches of independently and identically distributed (i.i.d) data samples. These batch learning algorithms distinguish between the processes of knowledge training and knowledge inference, and require restarting the full training process each time new data becomes available. Indeed, they need to access all of the training data within multiple epochs (offline training).

Continual learning (CL) considers the scenario of learning from a non i.i.d data stream where different data and tasks are presented to the model in a sequential manner. While humans can continually observe and learn new knowledge, DNNs have been shown to suffer from catastrophic forgetting (CF) - they are not able to perform well on previously seen data after being updated with recent data. Because their learned knowledge is stored in a single set of weights and learning new information modifies that set of weights, sometimes drastically, new learning frequently causes the networks to forget all, or most of, what they have previously learned.

Catastrophic forgetting was actively studied in the late 1980s and 1990s, mainly in neuro-scientific literature, and numerous solutions were adopted in an attempt to overcome the problem [41]. A more general problem, referred to as the stability-plasticity dilemma, where the trade-off between the ability of retaining previous knowledge and fast learning a new task, was studied as well [8]. CF is a direct consequence of the overlap of distributed representations. Indeed, highly distributed representations with significant interaction among them are able to generalize but suffer from CF while very local representations with little or no interaction among them do not suffer from CF but lack generalization ability or storage capacity.

Many early algorithms involved reducing the distributed nature of the network's internal representations, thereby creating relatively sparse representations to reduce the representational overlap. This was done in several ways, either by gradually nudging the network's internal representations into largely non-overlapping clusters [49] or by orthogonalizing the input representations [15, 33, 35]. Other solutions have involved the network learning "pseudo-patterns" that reflected the previously learned patterns [50]; using dual networks in which the old information was transferred to a second, independent network and then interleaved with the new information being learned [16, 4], creating an approximation of the error surface of the previously learned patterns and integrating that error surface with the error surface associated with the new patterns [18], and so on. See [17] for a review of early work on CF, dating from the early 1990's through the mid-2000's. A large number of recent continual learning methods have been proposed in the literature and many of them are built on those early solutions.

**Sparse Distributed Memory:** the standard Kanerva Sparse Distributed Memory (SDM) is a content-addressable memory model developed in the 1980's [27, 26]. In its development, SDM respected fundamental biological constraints and is considered to be a generalization of the original Hopfield network [28, 46]. Recent work [6] has shown mathematically that SDM closely approximates Attention, the core of widely used Transformer models. SDM was used as semi-distributed representations that allowed it to be robust to CF. In SDM, input content is stored in hyperspheres in a very high-dimensional space around hard locations that are largely, but not completely, independent and, thus, do not significantly interfere with one another. At the same time, the size of the hyperspheres is large enough that generalization is not significantly impaired. In this way, the representational overlap causing CF is avoided, while, at the same time, the generalization capacities of the system are not overly affected.

Taking into account these advantages of SDM, we propose a novel algorithm called Dynamic Sparse Distributed Memory (DSDM) that performs excellently on the most challenging setting of continual learning, namely, online task-free scenario. Figure 1 gives a brief overview and the difference between SDM and DSDM. Experiment results of DSDM on a number of image classification benchmarks under different scenarios demonstrate its excellent ability to eliminate catastrophic forgetting.

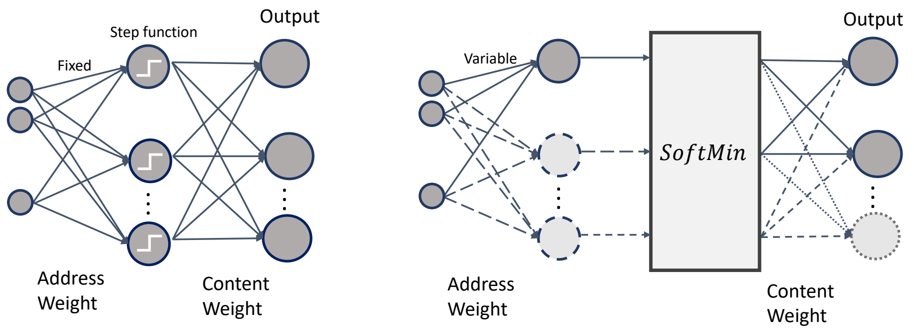

Fig. 1: SDM (left) vs. DSDM (right). While in SDM, the number of neurons is fixed, their addresses are randomly distributed and fixed, and the radius of reading and writing patterns are predefined, in DSDM all these factors are data-driven and dynamically learnt. The dashed/dotted lines/states mean that these connections and nodes will be added or removed as needed.

## 2 Related work

We first briefly discuss different scenarios of CL in Section 2.1 and then present several existing CL methods in Section 2.2. We provide an overview of the standard SDM on which our algorithm is based in Section 2.3.

### 2.1 Continual learning settings

Continual learning has been studied under different scenarios that are often divided into three categories of increasing difficulty [57]: **task, domain**, and **class-incremental**. In task-incremental learning, there is a single classification head assigned to each task, whereas in class-incremental learning, there is a single head for all tasks. Models doing domain-incremental learning use a single classification head and do not have to infer the identity of the task.

Online learning requires each image or item to be seen only once during learning, whereas in offline learning, the input data may be seen repeatedly, as is necessary. Additionally, continual learning can be task-based or task-free, depending on whether boundaries between different tasks are known or not. Recently, the **data-incremental learning** paradigm has been introduced [11] to facilitate learning from any data stream, without any assumption regarding task identity, task boundaries or the order of observing the data.

The work presented here studies the most challenging scenario of continual learning in an **online, completely task-free and class-incremental** (data incremental) setting where learning and evaluating can be carried out at any point of time.

## 2.2 Continual learning algorithms

Many continual learning methods have been developed in three major streams: regularization-based, architecture-based, and memory-based (or rehearsal-based) approaches.

**Regularization** This approach aims to preserve the knowledge acquired from previous tasks, while maintaining the plasticity of every node. Elastic Weight Consolidation (EWC) [31] estimates the importance of each of the weights on previous tasks using the Fisher information matrix. Weight changes are then based on the degree of importance of particular weights. Another regularization method penalizes each weight change according to a measure of synaptic intelligence (SI) [63]. An attempt is made to estimate the task specificity for each parameter and penalize changes of parameters with high task specificity. This ensures that the parameter stays close to its present value, a value that ensured good performance on learning the initial task.

Knowledge Distillation (KD) is an efficient manner of transferring the knowledge between networks. It was initially introduced for network compression by using the teacher-student mechanism [24] and widely adopted for CL methods [36, 52] and is considered to be one of regularization techniques.

However, it has been shown [57] that methods based on only regularization such as EWC and SI completely fail in a class-IL setting.

**Architecture** These methods are often based on architectural modifications. They introduce task-specific parameters that dynamically increase the capacity of the model by adding a new layer to the network, as in [58]. Other methods of this kind dynamically add new layers to the network [51]. Other strategies developed by [39]; [53] attempt to freeze weights that are judged by the system to be the most important to a particular task and that will, therefore, not be updated by future back-propagation. Similar strategies use a mask for each previous task to protect parameters during the backward pass or to choose which parameters to use during the forward pass.

**Memory-based approaches** The key idea is to choose a strategy to store specific exemplars. It can be a strategy as simple as random selection or a more complex method, as in [48]. Some models can replay the exemplars stored in memory with the stream of data from the current task or they can be used as a regularization term, as in GEM [38], or in A-GEM [10], by computing the scalar product of the loss gradient vectors of previous examples stored in memory and the current data gradient vector. The parameters of the model are then only updated if the scalar product is positive.

To overcome the necessity of an external memory to store data from previous tasks, as in memory-based approaches, certain architectures integrate a generative model able to generate exemplars from previous tasks. These models include the use of a Generative Adversarial Network (GAN) [19] as in [60, 44] or a Variational

AutoEncoder [30, 29] as in [56]. These methods are also called *Generative Replay* methods. The main drawback is that generative models are themselves prone to catastrophic forgetting. Moreover, methods which use a GAN to generate new samples, such as GDM [43] and ILUGAN [61], are both offline and task-based.

The proposed DSDM, a dynamic memory, can be considered to be a hybrid model that combines architecture-based and memory-based approaches.

### 2.3 Sparse Distributed Memory (SDM)

SDM is a content-addressable memory capable of both auto-associative and hetero-associative operation. It typically consisting of one million 1000-bit memory locations which are randomly distributed (i.e., fixed addresses) [26]. Using only a million locations out of possible $2^{1000}$ locations ($10^6 \ll 2^{1000}$), it is 'sparse'. SDM is mapped to a two-layer feed-forward neural network (without counting the input layer). The input units convey data patterns to the memory units (i.e., the address nodes). A pattern has two components: the pattern address is the vector representation of a memory point; and the pattern content. The weights to the address nodes represent a 'hard locations' and the weights to the output units store the memory contents.

**Storing patterns** When storing a pattern, the network computes the Hamming distance between the input address vector and each hard address. Address nodes within a preset Hamming distance will be activated (i.e., set to 1). In other words, if the fixed fan-in weight-vector associated with an address node is similar to the input vector, the address node will be activated; otherwise, it will remain inactive. The content weights that will be modified are those that fan-out from an active address nodes.

**Retrieving patterns** Reading is the operation of retrieving a data pattern from the memory at a particular address pattern. The input pattern is used to select a certain number of hard memory locations. The retrieved content is the average of the contents (i.e. fan-out weights) at the selected locations, followed by a fixed thresholding step.

## 3 Dynamic Sparse Distributed Memory

Conventional SDM networks have a fixed number of memory nodes whose 'hard' addresses are randomly distributed in a binary memory space. This tends to lower performance when dealing with correlated, i.e., nonrandom, input data [23]. In our algorithm, we start with an empty memory space, and new address nodes are incrementally added depending on the considered input patterns and the current state of memory space. In other words, we dynamically grow new address nodes as a function of the input. DSDM models the distribution of the input patterns much better than SDM where certain fixed neurons are too far from the

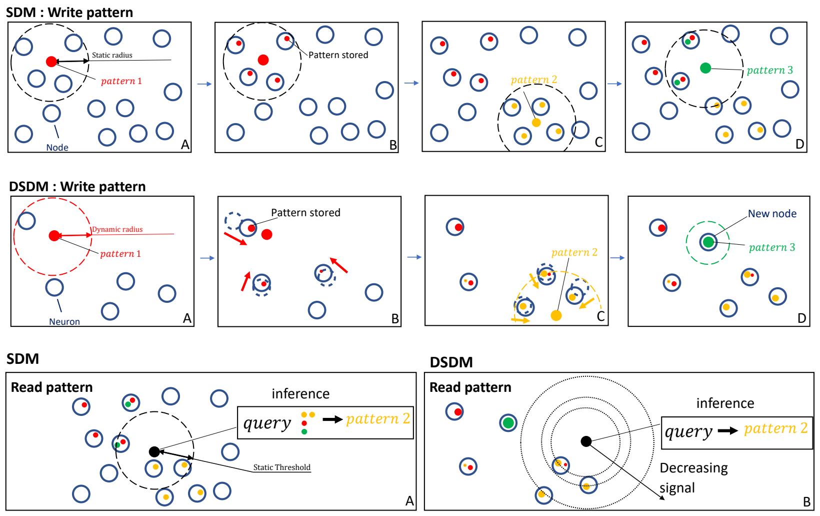

Fig. 2: The read and write operations of SDM and DSDM. **Top row, SDM write:** A+B+C. Two patterns are stored inside nearby fixed neurons within a fixed write radius; D. Writing a third pattern and neurons storing a superposition of multiple patterns. **Middle row, DSDM write:** A. Writing the first pattern that is close enough to its best matching unit with respect to a dynamic write radius; B. Neurons in space will move toward the pattern address while their content is sharpened with the pattern content; C. Writing the second pattern being close enough to its BMU; D. The third pattern is far from its BMU with respect to a dynamic radius, so a new neuron in memory is created. **Bottom row, Read operation:** A. SDM read: the query content reading from nearby neurons within a fixed radius using a majority vote; B. DSDM read: the query content is retrieved from multiple neurons according to its distance to the query address using a softmin function.

current input distribution and therefore are never activated, meaning that our memory has a higher storage capacity. We show experimentally in Section 4.2 that DSDM considerably outperforms SDM with far fewer neurons. Moreover, while a predefined Hamming distance is crucial in SDM, we propose to use a dynamic threshold, a so-called recursive temperature (RT) mechanism as a means of dynamically deciding whether a new memory unit is created.

In our work on image classification, we use DSDM network almost as a backend to a deep encoder and each address-content association consists of a feature vector, extracted from inputs by the encoder and a category label vector, coded in a one-hot manner.

**Algorithm 1** DSDM training

```
1: Init: Address matrix $A = \{a_1 = x_1\}$, content matrix $C = \{c_1 = y_1\}$ with capacity
   $Q$, $RT = 0; K = 1$;
2: Require: $\beta$, $\lambda_{RT}$, $\lambda$. $\lambda_{RT}$ and $\lambda$ are the different learning rates, one for updating
   the dynamic threshold, one for sharpening the address and content.
3: Input: Consider each sample $(x_t, y_t)$ from the data stream $\{(x_2, y_2), (x_3, y_3), ...\}$.
4: $BMU = \text{argmin}_{i \in [\![1,K]\!]} \|x_t - a_i\|$
5: $d_{BMU} \leftarrow \|x_t - a_{BMU}\|$
6: if $d_{BMU} > RT$ then
7:     $A \leftarrow A \cup (x_t)$
8:     $C \leftarrow C \cup (y_t)$
9:     $K \leftarrow K + 1$
10: else
11:     $A \leftarrow A + \lambda \times softmin(\|A - x_t\|/\beta) \times (A - x_t)$
12:     $C \leftarrow C + \lambda \times softmin(\|A - x_t\|/\beta) \times (C - y_t)$
13: end if
14: $RT \leftarrow \lambda_{RT} \times RT + (1 - \lambda_{RT}) \times \|x_t - a_{BMU}\|$
15: if $K > Q$ then # if memory is full
16:     LOF-based pruning
17: end if
```

## 3.1 Training phase

As can be seen in Figure 2, DSDM carries out two different types of learning, competitive and supervised learning, within its two layers respectively. The pseudo-code of DSDM training phase is described in Algorithm 1.

Our memory $M$ is represented by two matrices $A$ and $C$ that are initialized with the first input pattern, denoted by $(x_1, y_1)$, presented to the model. Our algorithm is online, each data sample is considered only once. It is also completely task-free, no assumption on task boundaries or the order of data presentation is required. Consider the pattern $\{(x_t, y_t)\}$ in the never-ending data stream $\{(x_2, y_2), (x_3, y_3), ...\}$ progressively presented to the model, we first find its nearest memory node, i.e., the best matching unit - BMU, in the memory space with respect to the location (address). For simplicity, we denote the index of BMU as "BMU":

$$\text{BMU} = \underset{i \in [\![1,K]\!]}{\text{argmin}} \|x_t - a_i\| \tag{1}$$

where $\|a_i - a_j\|$ is distance between two memory locations; $K$ is the number of neurons created so far; $a_{BMU}$ is then the nearest neuron of $x_t$; $\|x_t - a_{BMU}\|$ is hereafter called "to-BMU distance" ($d_{BMU}$).

In the literature, there exist several expandable architectures [62, 40, 14] and they compare the to-BMU distance or similar metric to a *pre-defined* threshold to decide whether a new node or layer is added. While other methods can add a new layer, our DSDM by design has only two layers (excluding the input layer) and we only add/remove nodes. Since the to-BMU distance changes depending on the order of observing data, using a static threshold cannot be optimal and this often leads to an inadequate trade-off between the memory performance and size. To overcome this problem, we define a dynamic threshold as a means of dynamically deciding on whether a new node is created. This is achieved by using an exponential moving average of the last to-BMU distances. We call this mechanism *recursive temperature* (RT) since its value tends to increase when DSDM starts to learn a new task and then gradually decreases (see section 4.6). The recursive temperature is dynamically updated as:

$$RT = \lambda_{RT} \times RT + (1 - \lambda_{RT}) \times \|x_t - a_{BMU}\| \tag{2}$$

where $\lambda_{RT}$ is the learning rate.

At this step, two possible cases can occur:

• If the to-BMU distance is lower than RT, meaning that the input sample is "close enough" to its BMU, the information for the input sample (address and content) will be distributedly stored across multiple nodes already created in memory. The address and content weights of these existing nodes are sharpened proportionally with respect to their distance to the considered input pattern's address, as follows:

$$A = A + \lambda \times softmin(\|A - x_t\|/\beta) \times (A - x_t) \tag{3}$$

$$C = C + \lambda \times softmin(\|A - x_t\|/\beta) \times (C - y_t) \tag{4}$$

where $\lambda$ is the learning rate for sharpening the address and content.

This update strategy is similar to that of competitive learning proposed by [28] to adapt models to the current data distribution.

• If the to-BMU distance is higher than RT, a new memory node is created and added into the memory space: $A \leftarrow A \cup (x_t)$, $C \leftarrow C \cup (y_t)$, $K \leftarrow K + 1$.

**Memory pruning** Besides using a recursive temperature mechanism to dynamically expand the memory space, we also propose using a density-based algorithm as a means of controlling memory size. In order to estimate the local density of patterns, we use in this work the Local Outlier Factor LOF algorithm [5], which is widely used for anomaly detection. However, we did not remove boundary samples as is done in LOF-based anomaly detection. Instead, we removed address nodes with high density, along with their associated content weights, until the desired number of address nodes has been reached. The intuition is that the removed patterns can still be retrieved by DSDM. We will continue to study different pruning techniques and threshold criteria.

## 3.2 Inference phase

Consider the input query with address $x_q \in \mathbb{R}^m$, to obtain the model output, we first compute its $L_2$ distance to every patterns in the memory space: $d(A, x_q) = [\|a_1 - x_q\|, \|a_2 - x_q\|, ..., \|a_K - x_q\|]$.

We then compute the activation vector of the first layer, denoted by $W = [w_1, w_2, ..., w_K]$, using a softmin function:

$$\forall j \in [\![1, K]\!], w_j = \frac{e^{-d_j/\beta}}{\sum_{i=1}^{K} e^{-d_i/\beta}} \tag{5}$$

where $\beta$ is a temperature parameter which has the same role as the fixed threshold in the original SDM. If we increase $\beta$, nodes being far from the query pattern will have higher contribution in the final decision. If $\beta$ decreases, these nodes being far from the query pattern will have less contribution. In the extreme case with $\beta$ close to zero, only the content of the nearest address to the query is considered.

The final output $y_q$ is finally obtained by:

$$y_q = \sum_{i=1}^{K} w_i * c_i \tag{6}$$

where $c_i$ is the content of $i^{th}$ node, and $*$ is the element-wise multiplication.

## 4 Experiments

### 4.1 Experiment setting

**Evaluation metrics.** We consider the commonly used evaluation metrics - namely, average accuracy (Avg) and last step accuracy (Last) where Avg is the average of accuracy obtained after learning of each task, and Last is the final performance once all tasks/classes have been learned one after the other. We ran each experiment five times on a computer with one NVIDIA RTX 2060 GPU and report the average Top-1 classification results.

**Datasets.** We evaluated our algorithm under an online class-incremental setting on five commonly used image classification datasets:

**MNIST dataset** [13] is a handwritten digit recognition dataset with 60k training samples and 10k testing samples.

**CIFAR-10 dataset** [32] contains 10 classes and each class is composed of 6k images. They are divided into 50k images for training and 10k for testing.

**CIFAR-100 dataset** [32] consists of 100 mutually exclusive classes. Within each class, there are 500 training images and 100 testing images.

**CUB-200 dataset** [59] includes 200 fine-grained bird species. There are 5994 images for training and 5794 images for testing. For each category, there are about 29-30 training images and 11-30 testing images.

**CORE-50 dataset** [37] consists of 164,866 128×128 color images. It has for each class around 2400 training images and 900 testing images. For the class-incremental setting, it is divided into nine tasks and has a total of 50 classes with 10 classes in the first task and 5 classes in the other eight tasks.

In our experiments on all five datasets, we randomly arrange the class order. For each dataset (except CORE-50), other authors often report their results on different incremental step values. For a dataset of 10 classes in total (MNIST or CIFAR-10), when the incremental step is set to 2, the dataset itself is divided into five tasks with two class labels each (with step = 2, we will call Split MNIST and Split CIFAR-10 respectively). Those algorithms have different performances for different step sizes since they are not completely task-free. By contrast, DSDM is data-incremental learning, i.e., online (training with batch size = 1) and completely task-free. In this data configuration, we invariably obtained excellent performance, meaning that our results do not depend on the order of observing data.

Our model has three hyperparameters, i.e., $\beta$, $\lambda_{RT}$, and $\lambda$, where $\beta$ is the temperature parameter of the softmin function being used during inference phase and has the same role as the fixed threshold in the original SDM, $\lambda_{RT}$ and $\lambda$ are the different learning rates, one for updating the dynamic threshold, the other for sharpening the addresses and contents. The model performed similarly for a wide range of parameter values: $\beta \in [0.5, 6]$, $\lambda, \lambda_{RT} \in [0.001, 0.01]$ when an encoder is used (see details in Sections 4.2, 4.3). Without an encoder (for the MNIST dataset only), we report the accuracies with $\beta = 0.5$, $\lambda = 0.0022, \lambda_{RT} = 0.0025$. In order words, for the moment these values are fixed. Future work will involve updating these values dynamically. One idea is to dynamically update $\beta$ as $K$ increases. We also consider having several BMUs, rather than only one BMU.

## 4.2 Advantage of DSDM over SDM

This section mainly aims at showing the advantage of DSDM over SDM. To this end, we consider the split MNIST benchmark and we do not use any encoder. The pattern address component of our memory is the original MNIST image vector, $x_t = I_t$. We modified the SDM to handle vectors of integers for multiple gray levels by replacing the Hamming distance with the city-block and $L_p$ distances. While it is theoretically shown in [1] that for SDM, the city-block distance can slightly outperform the $L_2$ distance (by 1-2%), it is not the same for our DSDM. Indeed, we evaluated DSDM with these two distances and found that DSDM performs better with the $L_2$ distance.

As can be seen in Table 1, DSDM significantly outperforms SDM (we set the DSDM model size to 5k that is the half of SDM). Their last accuracies of $94.0 \pm 0.2$ and $74.2 \pm 3.5$, respectively, are *surprisingly* good. DSDM achieved the SOTA performance without any encoder. Our associative memory model is much simpler than other competing algorithms and no gradient flow was even considered. By comparison, other methods in Table 1 (except the Ensemble method) used a multi-layer perceptron with two hidden layers of 400 nodes each and ReLU non-linearities were used in all hidden layers. To achieve $91.0 \pm 0.4$, the Ensemble algorithm used a variational autoencoder where the encoder half comprises two convolutional layers followed by two linear layers and the decoder has two linear layers followed by two transpose convolutional layers. The results of CURL [47], CN-DPM [34] and Ensemble [54] methods are taken from their original work while other ones are from [11].

| shallow | | deep | | | | | |
|---|---|---|---|---|---|---|---|
| SDM | DSDM | finetune | CURL[47] | CN-DPM[34] | Ensemble [54] | MIR [2] | CoPE[11] |
| $74.2 \pm 3.5$ | $\mathbf{94.0 \pm 0.2}$ | $19.7 \pm 0.05$ | $92.6 \pm 0.7$ | $93.2 \pm 0.1$ | $91.0 \pm 0.4$ | $93.2 \pm 0.4$ | $93.9 \pm 0.2$ |

Table 1: Last accuracy on split MNIST (step=2). Without any encoder and gradient-based update, the shallow DSDM model outperforms SOTA deep methods.

## 4.3 Comparison with online SOTA methods

We compare our method with existing online approaches including A-GEM [10], GSS [3], MIR [2], ASER [55], GDUMB [45] and Candidates Voting (CV) [21]. Similar to the very recent work of CV [21], we report the average and last accuracies on Split CIFAR-10 and CORE-50. A small version of ResNet-18 [22] pretrained on ImageNet [12] is used as the backbone for all of the methods compared. In DSDM, the address component of a pattern is the feature extracted from the image produced by ResNet-18: $x_t = ResNet18(I_t)$. CV and DSDM freeze the parameters of the backbone network, while others do not. We vary the buffer size $Q \in \{1k, 2k, 5k\}$ for comparisons.

On CORE-50, the dataset that is specifically designed for continual learning, DSDM significantly outperforms existing online approaches on all considered buffer sizes $Q$ with the single exception of CV with buffer sizes of 1k when measured by the average accuracy. As for the last accuracies, DSDM is on average 50% better than the second-best results. On the split CIFAR-10 benchmark, DSDM outperforms all other methods with a limited storage capacity $Q = 1k$ and its performance is essentially the same as the best algorithm (CV). Moreover, only our algorithm is completely task-free, the learning and evaluating can be carried out at any point in time. The CV algorithm is online but it is not task-free during training with the use of task boundaries.

Several methods like MIR and GSS in Table 2 can perform reasonably well in both task-based and task-free settings. Their performance in the latter is worse than in the former. Their performance on the CIFAR10 dataset under these settings are respectively 52.2% and 42.8% (-9.6% for MIR) and 56.7% and 38.4% (-18.3% for GSS).

## 4.4 Comparison with offline SOTA methods

We also compare our method with offline continual learning approaches that update the model in multiple epochs on the CIFAR-100 and CUB-100 benchmarks with different incremental steps. Similar to FearNet ([29]) and ILUGAN ([61]), Resnet-50 backbone pre-trained on ImageNet is used as the feature extractor in DSDM: $x_t = ResNet50(I_t)$. On these benchmarks, we set $Q = 3k$. We consider SOTA offline approaches including EWC [31], LWF [36], iCaRL [48], End-to-End [9], FearNet [29] and ILUGAN [61]. The reported results are from [61]. Although it is widely acknowledged that performance in the online scenario is worse than offline, our DSDM still outperforms those SOTA offline approaches

| Datasets | CIFAR-10, step = 2 | | | | | | CORE-50 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Buffer size $Q$ | 1k | | 2k | | 5k | | 1k | | 2k | | 5k | |
| Accuracy | Avg | Last | Avg | Last | Avg | Last | Avg | Last | Avg | Last | Avg | Last |
| A-GEM [10] | 43.0 | 17.5 | 59.1 | 38.3 | 74.0 | 59.0 | 20.7 | 8.4 | 21.9 | 10.3 | 22.9 | 11.5 |
| MIR [2] | 67.3 | 52.2 | 80.2 | 66.2 | 83.4 | 74.8 | 33.9 | 21.1 | 37.1 | 24.5 | 38.1 | 27.7 |
| GSS [3] | 70.3 | 56.7 | 73.6 | 56.3 | 79.3 | 64.4 | 27.8 | 17.8 | 31.0 | 18.9 | 31.8 | 21.1 |
| ASER [55] | 63.4 | 46.4 | 78.2 | 59.3 | 83.3 | 73.1 | 24.3 | 12.2 | 30.8 | 17.4 | 32.5 | 18.5 |
| GDUMB [45] | 73.8 | 57.7 | 83.8 | 72.4 | 85.3 | 75.9 | 41.2 | 23.6 | 48.4 | 32.7 | 54.3 | 41.6 |
| CV [21] | 76.0 | 62.9 | **84.9** | **74.1** | **86.1** | **77.0** | **45.1** | 26.5 | 50.7 | 34.5 | 56.3 | 43.1 |
| DSDM | **80.2** | **67.0** | 83.8 | 72.5 | 85.6 | 76.0 | 43.9 | **43.3** | **53.2** | **50.8** | **66.3** | **57.1** |
| DSDM/CV (%) | 107 | | 98 | | 99 | | 163 | | 147 | | 132 | |

Table 2: Average and last accuracy on CIFAR-10 (step=2) and CORE-50. Best results marked in bold; second best results are underlined. **Only our algorithm is completely task-free.** Several methods like MIR and GSS can perform reasonably well under both task-based and task-free settings. We report here their best performance when task boundaries are known. The lines from "A-GEM" to "CV" are from [21].

with important margins, as can be seen in Table 3(a). Table 3(b) presents the training batch size and epoch number of different algorithms and clearly shows the advantages of DSDM.

| Datasets | CIFAR-100 | | CUB-200 | |
|---|---|---|---|---|
| Step | 2 | 5 | 2 | 5 |
| EWC [31] | 15.4 | 17.9 | 15.9 | 18.2 |
| LwF [36] | 32.8 | 37.9 | 29.8 | 36.1 |
| iCaRL [48] | 52.8 | 57.2 | 44.3 | 49.6 |
| End-to-End [9] | 50.3 | 49.7 | 44.5 | 49.9 |
| FearNet [29] | 56.9 | 62.5 | 47.8 | 52.7 |
| ILUGAN [61] | 58.0 | 63.1 | 49.7 | 54.9 |
| **DSDM** | **63.3** | **63.2** | **55.5** | **55.2** |

(a)

| Datasets | CIFAR-100 | | CUB-200 | |
|---|---|---|---|---|
| Training | BS | Ep | BS | Ep |
| iCaRL | 450 | 70 | 450 | 70 |
| End-to-End | 128 | 70 | 128 | 70 |
| FearNet | 450 | 250 | 200 | 100 |
| ILUGAN | | 90 | | 90 |
| CV | 10 | 1 | 10 | 1 |
| DSDM | 1 | 1 | 1 | 1 |

(b)

Table 3: (a) Comparison of average accuracy on CIFAR-100 and CUB-200. (b) Comparison of training batch size (BS) and epochs (Ep). **Only DSDM can carry out learning and evaluating at any point in time.**

### 4.5 Completely task-free and few-shot settings

**Completely task-free:** As mentioned above, other authors often reported different results with different incremental step sizes since their algorithms distinguish tasks that have clear task boundaries, at least during the training

| **Dataset** | **CIFAR-10** | | | **CIFAR-100** | | |
|---|---|---|---|---|---|---|
| step | 1 | 2 | Gaussian | 1 | 5 | Gaussian |
| CoPE [11] | – | 48.90 | – | – | 21.60 | – |
| CN-DPM [34] | – | 45.21 | – | – | 20.10 | – |
| Vanilla classifier [54] | 10.6±2.2 | 23.2±4.8 | 11.4±2.4 | 1.0±0.2 | 3.8±0.5 | 6.9±2.9 |
| Ensemble [54] | 78.3±0.4 | 79.0±0.4 | 50.1±9.5 | 54.1±0.5 | 55.3±0.4 | 39.0±1.4 |
| **DSDM+CNN, low data** | <u>79.4±0.5</u> | <u>79.6±0.5</u> | <u>78.7±1.1</u> | <u>54.9±1.4</u> | <u>55.3±1.3</u> | <u>55.5 ±1.2</u> |
| **DSDM+ViT, low data** | **85.5±0.7** | **85.6±0.6** | **84.9±0.6** | **61.1±0.5** | **60.8±0.9** | **61.4 ±1.1** |

Table 4: Comparison of last accuracy of different methods under the challenging scenarios. Best results marked in bold; second best results are underlined. Our algorithms works well on limited-data regimes.

phase. By contrast, our model operates in the completely task-free setting (DSDM achieved similar performances as can be seen in Tables 1, 2). We demonstrate this capability by evaluating DSDM under the Gaussian schedule benchmarks, recently proposed in [54]. A schedule defines how a data distribution evolves over training. In the Gaussian schedule, there are no task boundaries. Each label appears in the data with a probability that follows a Gaussian distribution, and each label's probability peaks at a different time.

**Few-shot CL:** We further study the data efficiency of our model. We report the results on CIFAR-10 and CIFAR-100 when using only 10% and 30% of the training samples from these datasets respectively. We use here two different feature extractors. We use two different feature extractors, ResNet50 pretrained on ImageNet in a supervised manner and a vision transformer encoder pretrained with self-supervised learning (SSL) [7]). (The simplest version ViT8 was used). We denote these methods as 'DSDM + CNN' and 'DSDM + ViT' respectively. For 'DSDM + CNN', we used a buffer size $Q = 2k, 6k$ for CIFAR-10 and CIFAR-100, respectively. For 'Ours ViT', we used a buffer size $Q = 1k, 3k$ for CIFAR-10 and CIFAR-100, respectively.

Table 4 compares the last accuracies under different scenarios. To the best of our knowledge, only the Ensemble architecture using a fixed ResNet50 pretrained in a SSL manner on ImageNet reports its performance with a Gaussian schedule (other results of Table 4 come from [54]). DSDM outperforms Ensemble while requiring much less data. Under the challenging scenario of a Gaussian schedule, DSDM's gain is highly significant. The performance accuracies of 'DSDM + ViT' are respectively 69% and 57% better than those of Ensemble on CIFAR-10 and CIFAR-100, respectively.

## 4.6 Impact of a dynamic threshold

This section shows the importance of using a dynamic threshold. Figure 3 shows the evolution of recursive temperature and to-BMU distances on the Core-50 training dataset. The to-BMU distances change when new data is seen.

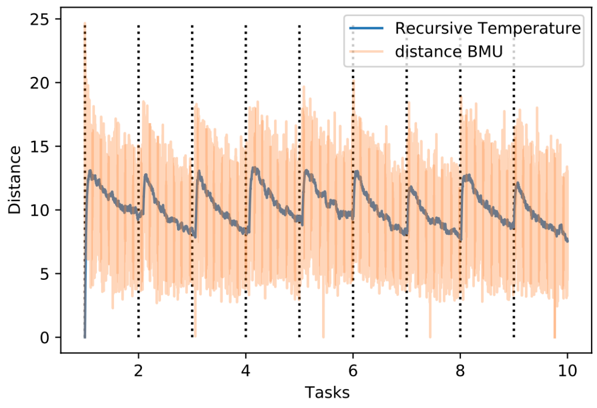

Fig. 3: The evolution of recursive temperature and to-BMU distance on the training set of Core-50

## 5 Conclusion

This work introduces DSDM, an online and completely task-free algorithm called DSDM for class-incremental learning. DSDM is an associative content-addressable memory model that evolves dynamically and continually to model the distribution of non-stationary data streams. With DSDM, learning and evaluating can be carried out at any point in time. DSDM outperforms SOTA methods on many image classification benchmarks, including MNIST, CIFAR-10, CIFAR-100, CUB-200 and Core-50 under different challenging scenarios, even without an encoder front-end and with limited training data.

Modifying or eliminating the use of DSDM's fixed encoder is currently under study. [20] has shown that an SSL encoder can be trained in online learning with random images. Moreover, [42] showed that features learned by an SSL encoder on a subset of the visual experiences of developing children led to high accuracy in non-trivial downstream categorization tasks. Further, the capacity to generalize under these circumstances was shown in [7]. This potentially means that our algorithm could generalize to new data that is very different from the initial pre-training data. In future work we will explore training a combination of an encoder and our DSDM architecture in an online and continual manner with a scenario corresponding to that of developing children. Indeed, we could first learn good representation features with SSL, followed by the continual learning phase with supervised learning. This scenario would arguably simulate real human learning more realistically than most current CL scenarior that start directly with incremental learning without any pretraining. We will, in addition, study how to apply DSDM to other tasks, such as regression or anomaly detection, as in [25].

## References

1. Aggarwal, C.C., Hinneburg, A., Keim, D.A.: On the Surprising Behavior of Distance Metrics in High Dimensional Space. In: Van den Bussche, J., Vianu, V. (eds.) 8th International Conference on Database Theory. pp. 420–434. Springer (2001)
2. Aljundi, R., Belilovsky, E., Tuytelaars, T., Charlin, L., Caccia, M., Lin, M., Page-Caccia, L.: Online Continual Learning with Maximal Interfered Retrieval. In: Advances in Neural Information Processing Systems. vol. 32 (2019)
3. Aljundi, R., Lin, M., Goujaud, B., Bengio, Y.: Gradient based sample selection for online continual learning. In: Advances in Neural Information Processing Systems. vol. 32 (2019)
4. Ans, B., Rousset, S., French, R.M., Musca, S.: Self-refreshing memory in artificial neural networks: learning temporal sequences without catastrophic forgetting. Connection Science **16**(2), 71–99 (Jun 2004)
5. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: Identifying Density-Based Local Outliers p. 12
6. Bricken, T., Pehlevan, C.: Attention Approximates Sparse Distributed Memory. Advances in Neural Information Processing Systems (Nov 2021)
7. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging Properties in Self-Supervised Vision Transformers. 2021 IEEE/CVF International Conference on Computer Vision (ICCV) (May 2021)
8. Carpenter, G., Grossberg, S.: ART 2: self-organization of stable category recognition codes for analog input patterns, appl. opt. 26, 4919-4930 (1987)
9. Castro, F.M., Marín-Jiménez, M.J., Guil, N., Schmid, C., Alahari, K.: End-to-End Incremental Learning. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Proceedings of the European conference on computer vision (ECCV) (2018)
10. Chaudhry, A., Ranzato, M., Rohrbach, M., Elhoseiny, M.: Efficient Lifelong Learning with A-GEM. In: International Conference on Learning Representations (Sep 2018)
11. De Lange, M., Tuytelaars, T.: Continual Prototype Evolution: Learning Online From Non-Stationary Data Streams (2021)
12. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition (Jun 2009)
13. Deng, L.: The MNIST Database of Handwritten Digit Images for Machine Learning Research. IEEE Signal Processing Magazine **29**(6), 141–142 (Nov 2012)
14. Fahlman, S.E., Lebiere, C.: The cascade-correlation learning architecture. In: Touretzky, D.S. (ed.) Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]. pp. 524–532. Morgan Kaufmann (1989), http://papers.nips.cc/paper/207-the-cascade-correlation-learning-architecture
15. French, R.M.: Dynamically Constraining Connectionist Networks to Produce Distributed, Orthogonal Representations to Reduce Catastrophic Interference. Proceedings of the 16th Annual Cognitive Science Society Conference (Aug 1994)
16. French, R.M.: Pseudo-recurrent Connectionist Networks: An Approach to the 'Sensitivity-Stability' Dilemma. Connection Science (Dec 1997)
17. French, R.M.: Catastrophic forgetting in connectionist networks. Trends in Cognitive Sciences (Apr 1999)
18. French, R.M., Chater, N.: Using Noise to Compute Error Surfaces in Connectionist Networks: A Novel Means of Reducing Catastrophic Forgetting. Neural Computation (Jul 2002)

19. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative Adversarial Networks. arXiv:1406.2661 (Jun 2014)
20. Goyal, P., Caron, M., Lefaudeux, B., Xu, M., Wang, P., Pai, V., Singh, M., Liptchinsky, V., Misra, I., Joulin, A., Bojanowski, P.: Self-supervised Pretraining of Visual Features in the Wild. arXiv:2103.01988 (Mar 2021)
21. He, J., Zhu, F.: Online Continual Learning Via Candidates Voting. In: 2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) (Jan 2022)
22. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Jun 2016)
23. Hely, T.A., Willshaw, D.J., Hayes, G.M.: A new approach to Kanerva's sparse distributed memory. IEEE transactions on neural networks **8**(3), 791–794 (1997). https://doi.org/10.1109/72.572115
24. Hinton, G., Vinyals, O., Dean, J.: Distilling the Knowledge in a Neural Network. arXiv:1503.02531 (Mar 2015)
25. Jezequel, L., Vu, N., Beaudet, J., Histace, A.: Efficient anomaly detection using self-supervised multi-cue tasks. CoRR **abs/2111.12379** (2021), https://arxiv.org/abs/2111.12379
26. Kanerva, P.: A cerebellar-model associative memory as a generalized random-access memory. In: Thirty-Fourth IEEE Computer Society International Conference: Intellectual Leverage (Feb 1989)
27. Kanerva, P.: Sparse distributed memory and related models. Tech. Rep. NASA-CR-190553 (Apr 1992), keeler
28. Keeler, J.: Capacity for Patterns and Sequences in Kanerva' s SDM as Compared to Other Associative Memory Models. In: Neural Information Processing Systems. American Institute of Physics (1988)
29. Kemker, R., Kanan, C.: FearNet: Brain-Inspired Model for Incremental Learning. In: International Conference on Learning Representations (Feb 2018)
30. Kingma, D.P., Welling, M.: Auto-Encoding Variational Bayes. arXiv:1312.6114 (May 2014)
31. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., Hadsell, R.: Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences **114**(13), 3521–3526 (Mar 2017)
32. Krizhevsky, A.: Learning Multiple Layers of Features from Tiny Images (2009)
33. KRUSCHKE, J.K.: Human Category Learning: Implications for Backpropagation Models. Connection Science (Jan 1993)
34. Lee, S., Ha, J., Zhang, D., Kim, G.: A neural dirichlet process mixture model for task-free continual learning. In: 8th ICLR 2020. OpenReview.net (2020), https://openreview.net/forum?id=SJxSOJStPr
35. Lewandowsky, S.: Gradual unlearning and catastrophic interference: A comparison of distributed architectures. In: Relating theory and data: Essays on human memory in honor of Bennet B. Murdock, pp. 445–476. Lawrence Erlbaum Associates, Inc, Hillsdale, NJ, US (1991)
36. Li, Z., Hoiem, D.: Learning without forgetting: 14th European Conference on Computer Vision, ECCV 2016. Computer Vision - 14th European Conference, ECCV 2016, Proceedings pp. 614–629 (2016)
37. Lomonaco, V., Maltoni, D.: CORe50: a New Dataset and Benchmark for Continuous Object Recognition. In: CoRL (2017)

38. Lopez-Paz, D., Ranzato, M.A.: Gradient Episodic Memory for Continual Learning. In: Advances in Neural Information Processing Systems (2017)
39. Mallya, A., Lazebnik, S.: PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (Jun 2018)
40. Marsland, S., Shapiro, J., Nehmzow, U.: A self-organising network that grows when required. Neural Networks (Oct 2002)
41. McCloskey, M., Cohen, N.J.: Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In: Bower, G.H. (ed.) Psychology of Learning and Motivation, vol. 24 (Jan 1989)
42. Orhan, A.E., Gupta, V.V., Lake, B.M.: Self-supervised learning through the eyes of a child. Advances in Neural Information Processing Systems (Dec 2020)
43. Ostapenko, O., Puscas, M., Klein, T., Jähnichen, P., Nabi, M.: Learning to Remember: A Synaptic Plasticity Driven Framework for Continual Learning. arXiv:1904.03137 [cs] (Dec 2019), http://arxiv.org/abs/1904.03137, arXiv: 1904.03137
44. Ostapenko, O., Puscas, M.M., Klein, T., Jähnichen, P., Nabi, M.: Learning to remember: A synaptic plasticity driven framework for continual learning. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. pp. 11321–11329. Computer Vision Foundation / IEEE (2019). https://doi.org/10.1109/CVPR.2019.01158
45. Prabhu, A., Torr, P.H.S., Dokania, P.K.: GDumb: A Simple Approach that Questions Our Progress in Continual Learning. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (eds.) Computer Vision – ECCV 2020, vol. 12347, pp. 524–540 (2020)
46. Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Adler, T., Gruber, L., Holzleitner, M., Pavlović, M., Sandve, G.K., Greiff, V., Kreil, D., Kopp, M., Klambauer, G., Brandstetter, J., Hochreiter, S.: Hopfield Networks is All You Need. arXiv:2008.02217 (Apr 2021)
47. Rao, D., Visin, F., Rusu, A.A., Teh, Y.W., Pascanu, R., Hadsell, R.: Continual unsupervised representation learning. CoRR **abs/1910.14481** (2019), http://arxiv.org/abs/1910.14481
48. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: iCaRL: Incremental Classifier and Representation Learning. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Jul 2017)
49. R.M, F.: Semi-distributed Representations and Catastrophic Forgetting in Connectionist Networks. Connection Science (Jan 1992)
50. ROBINS, A.: Catastrophic Forgetting, Rehearsal and Pseudorehearsal. Connection Science (Jun 1995)
51. Roy, D., Panda, P., Roy, K.: Tree-CNN: A Hierarchical Deep Convolutional Neural Network for Incremental Learning. Neural Networks (Sep 2019)
52. Schwarz, J., Luketina, J., Czarnecki, W.M., Grabska-Barwinska, A., Teh, Y.W., Pascanu, R., Hadsell, R.: Progress & Compress: A scalable framework for continual learning. Proceedings of the 35th International Conference on Machine Learning (Jul 2018)
53. Serrà, J., Surís, D., Miron, M., Karatzoglou, A.: Overcoming catastrophic forgetting with hard attention to the task. International Conference on Machine Learning (May 2018)
54. Shanahan, M., Kaplanis, C., Mitrović, J.: Encoders and Ensembles for Task-Free Continual Learning. arXiv:2105.13327 (Oct 2021)

55. Shim, D., Mai, Z., Jeong, J., Sanner, S., Kim, H., Jang, J.: Online Class-Incremental Continual Learning with Adversarial Shapley Value. Proceedings of the AAAI Conference on Artificial Intelligence (May 2021)
56. van de Ven, G.M., Siegelmann, H.T., Tolias, A.S.: Brain-inspired replay for continual learning with artificial neural networks. Nature Communications (Dec 2020)
57. van de Ven, G.M., Tolias, A.S.: Three scenarios for continual learning. arXiv:1904.07734 (Apr 2019)
58. Wang, Y.X., Ramanan, D., Hebert, M.: Growing a Brain: Fine-Tuning by Increasing Model Capacity. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (Jul 2019)
59. Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., Perona, P.: Caltech-ucsd birds 200 (2010)
60. Wu, C., Herranz, L., Liu, X., wang, y., van de Weijer, J., Raducanu, B.: Memory Replay GANs: Learning to Generate New Categories without Forgetting. In: Advances in Neural Information Processing Systems (2018)
61. Xiang, Y., Fu, Y., Ji, P., Huang, H.: Incremental Learning Using Conditional Adversarial Networks. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV) (Oct 2019)
62. Yoon, J., Yang, E., Lee, J., Hwang, S.J.: Lifelong Learning with Dynamically Expandable Networks. In: International Conference on Learning Representations (Feb 2018)
63. Zenke, F., Poole, B., Ganguli, S.: Continual Learning Through Synaptic Intelligence. In: Proceedings of the 34th International Conference on Machine Learning (Jul 2017)