Task-Free Continual Learning

Rahaf Aljundi * Klaas Kelchtermans* Tinne Tuytelaars
KU Leuven, ESAT-PSI, Belgium
firstname.lastname@esat.kuleuven.be

Abstract

Methods proposed in the literature towards continual deep learning typically operate in a task-based sequential learning setup. A sequence of tasks is learned, one at a time, with all data of current task available but not of previous or future tasks. Task boundaries and identities are known at all times. This setup, however, is rarely encountered in practical applications. Therefore we investigate how to transform continual learning to an online setup. We develop a system that keeps on learning over time in a streaming fashion, with data distributions gradually changing and without the notion of separate tasks. To this end, we build on the work on Memory Aware Synapses, and show how this method can be made online by providing a protocol to decide i) when to update the importance weights, ii) which data to use to update them, and iii) how to accumulate the importance weights at each update step. Experimental results show the validity of the approach in the context of two applications: (self-)supervised learning of a face recognition model by watching soap series and learning a robot to avoid collisions.

1. Introduction

In machine learning, one of the most basic paradigms is to clearly distinguish between a training and testing phase. Once a model is trained and validated, it switches to a test mode: the model gets frozen and deployed for inference on previously unseen data, without ever making changes to the model parameters again. This setup assumes a static world, with no distribution shifts over time. Further, it assumes a static task specification, so no new requirements in terms of output (e.g. novel category labels) or new tasks added over time. Such strong division between training and testing makes it easier to develop novel machine learning algorithms, yet is also very restrictive.

Inspired by biological systems, the field of incremental learning, also referred to as continual learning or lifelong learning [24, 34, 37], aims at breaking this strong barrier between the training and testing phase. The goal is to develop algorithms that do not stop learning, but rather keep updating the model parameters over time. This holds the promise of a system that gradually accumulates knowledge, reaching increasingly better accuracy and better coverage as time passes. However, it is practically not possible to store all previous data - be it due to storage constraints or for privacy reasons. Yet updating parameters based only on recent data introduces a bias towards that data and a phenomenon known as catastrophic interference, in other words degrading performance on older data [8, 30].

To make progress in this direction, several works have opted for a specific experimental setup, consisting of a sequence of distinct tasks, learned one after the other. Each time, only the data for the 'current' task is available for training. We refer to this as *task-based sequential learning*. Training a shared model one task at a time has led to significant progress and new insights towards continual learning, such as different strategies for preserving the knowledge of previous tasks [19, 13, 1, 17]. However, the methods developed in this specific setup all too often depend on knowing the task boundaries. These boundaries indicate good moments to consolidate knowledge, namely after learning a task. Moreover, data can be shuffled within a task so as to guarantee i.i.d. data. In an online setting, on the other hand, data needs to be processed in a streaming fashion and data distributions might change gradually.

In this work, we aim at overcoming this requirement of hard task boundaries. In particular, we investigate how methods proposed for task-based sequential learning can be generalized to an online setting. This requires a protocol to determine when to consolidate knowledge. Moreover, we investigate the effect of keeping a small buffer with difficult samples. For the latter, we take inspiration from the field of reinforcement learning, namely experience replay [22], although using much smaller replay buffers, unlike very recent work of Rolnick et al. [31].

Task-based sequential learning has mostly been studied for image classification [19, 2, 17, 39, 28]. Whenever the learner arrives at a new task, that is when learning on the

*Rahaf Aljundi and Klaas Kelchtermans contributed equally to this work and are listed in alphabetical order.

previous task has converged, a standard procedure is to extend the output layer of the network with additional 'heads' for each of the new task's categories. Instead, the output of our network is fixed. In our first application, learning to recognize faces, we cope with a varying number of categories by using an embedding rather than class predictions. In our second application, learning a lightweight robot to navigate without collisions, it is not the output labels that change over time but rather the environment. For both applications, data is processed in a streaming fashion. This is challenging, since the data is not i.i.d. causing samples within one batch to be unbalanced.

The contributions of this paper are as follows: i) We are the first to extend task-based sequential learning to free and unknown task boundaries in an online continual learning scenario; ii) We develop protocols to integrate an importance weight regularizer, MAS, in this online continual learning setting; iii) Our experiments on face recognition from TV series and on monocular collision avoidance prove the effectiveness of our method in handling the distribution changes in the streaming data and stabilizing the online learning behaviour, resulting in knowledge accumulation rather than catastrophic interference and improved performance in all the test cases.

In the following we discuss related work (section 2). We then describe our online continual learning approach in section 3. We validate our system in the experimental section 4 and end with discussion and conclusion in section 5.

2. Related Work

Online Learning: Whereas in traditional offline learning, the entire training data has to be made available prior to learning the task, on the contrary online learning studies learning algorithms that learn to optimize predictive models over a stream of data instances sequentially. We refer to [5, 33] for surveys and overviews on the topic.

A first set of online learning algorithms consists of different techniques designed to learn a linear model [9, 6, 36, 12]. Online learning with kernels [14] extends this line of work to non-linear models, but the models remain shallow and their performance lags behind the modern deep neural networks. Unfortunately, attempts towards online learning of neural networks suffer from issues like convergence, catastrophic interference and more. Some recent works include [32, 27], who both start from a small network and then adapt the capacity by adding more neurons as new samples arrive, while for online deep metric learning, [18] proposed a method based on stacking multiple metric layers.

In terms of applications, the work of Pernici *et al.* [26, 25] is similar to our first application scenario. They learn face identities in a self-supervised fashion via temporal consistency. They start from the VGG face detector and descriptor, and use a memory of detected faces. In contrast, we start from a much weaker pretrained model (not face-specific), and update the model parameters over time while they do not.

A joint problem in continual and online learning is catastrophic interference [21, 8] which is the severe forgetting of previous samples when learning new ones. This phenomenon manifests itself at different scales: in online learning it happens while learning samples with different patterns than previous ones; in the traditional setting of continual learning it happens over a sequence of tasks.

Continual Learning: In [11], Hsu *et al.* classify the studied scenarios for continual learning into incremental task learning, incremental domain learning and incremental class learning. They argue that more attention should go to the last two - i.e. to methods that do not require to know the task identity, since that is the case encountered in most practical scenarios.

Yet as indicated before, most methods to date follow the task-based sequential learning setup. This includes various regularization-based methods, such as Elastic Weight Consolidation [13], Synaptic Intelligence [39] and Memory Aware Synapses [1]. These methods estimate importance weights for each model parameter and penalize changes to parameters deemed important for previous tasks. We will discuss how to extend one of them to the online setting later. Note that, while Synaptic Intelligence computes the importance weights in an online fashion, it still waits until the end of a task before updating the losses, so like the other methods it depends on knowing the task boundaries. Incremental Moment Matching [17] builds on similar ideas, yet stores different models for different tasks and merges them only at the very end. As such, it is unclear how this could be extended to an online, task-free setting. Also related is the work on Dynamically Expandable Networks [38]. They exploit the relatedness between the new task and previously learned tasks to determine which neurons can be reused and add new neurons to account for the new knowledge.

Next there are several data-driven methods such as Learning without Forgetting [19] or Encoder-based Lifelong Learning [28]. With a separate knowledge distillation loss term for previous tasks, it's again unclear how they could be applied without knowing the task identity.

Other methods use an episodic memory, such as iCARL (incremental Classifier and Representation Learning) [29] and Memory Based Parameter Adaptation [35]. A special mention here goes to Gradient Episodic Memory for Continual Learning [20], as it moves a step forward towards the online setting: it assumes that the learner receives examples one by one but simplifies the scenario to locally i.i.d. drawn samples from a task distribution. Moreover, it still assumes that a task identifier is given. Like the buffer we use, they use an episodic memory for each task consisting of recently seen examples. A buffer from which recent

data can be reused for training is similar to the concept of a replay buffer often used in Deep Reinforcement Learning (DRL). However a crucial difference is that in both old and recent DRL works the replay buffer typically contains up to 1M samples corresponding to over 100 days of experience [22, 10]. Here, we want to keep the algorithm more online by using a buffer of up to 100 samples only.

A common DRL technique, known as "prioritized sweeping", is to sample experiences with a large error more often than others [23]. In a similar fashion we propose "prioritized keeping" where a hard buffer drops the easy samples first rather than the oldest.

3. Method

Our goal is to design a training method for task-free online continual learning. Task-based sequential learning methods assume that data comes in tasks, with tasks boundaries identified, so the training procedure can be divided in consecutive phases. In between the training phases, when training has stabilized, the continual learning method updates its meta-knowledge on how to avoid forgetting previous tasks. However, in the case of online learning where data is streaming and the distribution is shifting gradually, it is unclear whether these methods can be applied and how.

After studying a couple of methods mentioned above, we identified Memory Aware Synapses (MAS) [1] as the most promising method in this respect. It enjoys the following favorable characteristics. 1) *Static storage requirement*: it only stores an importance weight for each parameter in the network avoiding an increase of memory consumption over time; 2) *Task agnostic*: it can be applied to any task and is not limited to classification. In particular, we can use it with an embedding as output, avoiding the need to add extra 'heads' for new outputs over time; 3) *Fast*: it only needs one backward pass to update the importance weights. During training, the gradients of the imposed penalty are simply the change that occurs on each parameter weighted by its importance. Therefore, the penalty gradients can be added locally and do not need a backpropagation step; 4) *top performance*: MAS shows superior performance to other importance weight regularizers [1, 11]. In order to deploy MAS in an online continual learning scenario, we need to determine i) when to update the importance weights, ii) which data to use to update the importance weights, and iii) how to accumulate the importance weights at each update step.

We first introduce the considered online continual learning setup, then explain MAS and our training procedure under this setup.

Setup: We assume an infinite stream of data and a supervisory or self-supervisory signal that is generated based on few consecutive samples. At each time step s, the system receives a few consecutive samples along with their generated labels $\{x_k, y_k\}$ drawn non i.i.d from a current distribution D_t. Moreover, the distribution D_t could itself experience sudden or gradual changes from D_t to D_{t+1} at any moment. The system is unaware of when these distribution changes are happening. The goal is to continually learn and update a function F that minimizes the prediction errors on previously seen and future samples. In other words, it aims at continuously updating and accumulating knowledge. Given an input model with parameters θ, the system at each time step reduces the empirical risk based on the recently received samples and a small buffer B composed of updated hard samples X_B. The learning objective of the online system is:

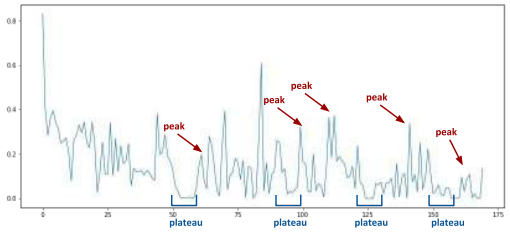


Figure 1: By detecting plateaus and peaks in the loss surface our method decides when to update the importance weights, corresponding to the Big Bang Theory experiment, see section 4.2; x-axis represents update steps

$$\min_{\theta} \mathcal{L}(F(X;\theta), Y) + \mathcal{L}(F(X_B;\theta), Y_B) \tag{1}$$

Due to the strong non-i.i.d conditions and the very low number of samples used for the gradient step, the system is vulnerable to catastrophic interference between recent samples and previous samples and faces difficulty in accumulating the knowledge over time.

Memory Aware Synapses (MAS) [1]: In a traditional task-based sequential learning setting, MAS works as follows. After each training phase (task), the method estimates an importance weight for each network parameter indicating the importance of this parameter to the previously learned task. To estimate the importance, MAS computes the sensitivity of the learned function to the parameters changes.

$$F(x_k;\theta+\delta) - F(x_k;\theta) \approx \sum_i g_i(x_k)\delta_i \tag{2}$$

$$\Omega_i = \frac{1}{N}\sum_{k=1}^{N} \| g_i(x_k) \| \tag{3}$$

where $\{x_k\}$ are the N samples from the previous task, δ_i is a small change to model parameter θ_i and $g_i(x_k) = \frac{\partial F(x_k)}{\partial \theta_i}$. Ω_i is the importance weight of parameter θ_i. When learning a new task, changes to important parameters are penalized:

$$L(\theta) = L_n(\theta) + \frac{\lambda}{2}\sum_i \Omega_i(\theta_i - \theta_i^*)^2 \tag{4}$$

with θ^* the parameters values at the time of importance weight estimation, i.e. the optimal parameters for the previous task in the traditional sequential setup. $L_n(\theta)$ is the

Algorithm 1 Online Continual Learning

Input:l_th,std_th,passes
Initialize: B={},{Ω_i}=0, check_plateau=False
Receive: $\{x, y\}$ ▷ K few consecutive samples
for pass in passes **do**
  $loss = \frac{1}{K}\sum_k \mathcal{L}(F(x_k;\theta), y_k)$
  $Hloss = \mathcal{L}(F(X_B;\theta), Y_B)$
  Backprob($Hloss + loss$), Update(θ)
  if $pass = 0$ **then**
    Update_Loss_Window ($Hloss + loss$)
  end if
end for
Update_Buffer ($\{x, y\}, loss, Hloss$)
if check_plateau and $\mu(l\_window) < l\_th$ and $\sigma(l\_window) < std\_th$ **then**
  Update(Ω, B)
  $l\_window = \{\}$, check_plateau=False
  $plt\_\mu = \mu(l\_window)$, $plt\_\sigma = \sigma(l\_window)$
end if
if $\mu(l\_window) > plt\_\mu + plt\_\sigma$ **then**
  check_plateau=True
end if

loss for the new task. After each task the newly estimated Ω_i are accumulated to the previous estimates.

When to update importance weights: in the case of a task-based sequential learning setting where tasks have predefined boundaries, importance weights are updated after each task, when learning has converged. In the online case, the data is streaming without knowledge of a task's start or end (i.e. when distribution shifts occur). So we need a mechanism to determine when to update the importance weights. For this, we look at the surface of the loss function.

By observing the loss, we can derive some information about the data presented to the system. When the loss decreases, this indicates that the model has learned some meaningful new knowledge from those seen samples. Yet the loss does not systematically decrease all the time. When new samples are received that are harder or contain different objects or input patterns than what was presented to the learner before, the loss may increase again. In those cases, the model has to update its knowledge, while minimally interfering with what has been learned previously.

We can conclude that plateaus in the loss function indicate stable learning regimes, where the model is confidently predicting the current labels, see Figure 1. Whenever the model is in such a stable area, it's a good time to consolidate the knowledge by updating the importance weights. This way, we identify the parameters that are important for the currently acquired knowledge. When learning new, "different" samples the model will then be encouraged to preserve this knowledge. This should allow the model to accumulate knowledge over time rather than replace previously learned bits of knowledge.

Detecting plateaus in the loss surface: to detect these plateaus in the loss surface, we use a sliding window over consecutive losses during training. We monitor the mean and the variance of the losses in this window and trigger an importance weight update whenever they are both lower than a given threshold. We do not keep re-estimating importance weights: we only re-check for plateaus in the loss surface after observing a peak. Peaks are detected when the window loss mean becomes higher than 85% of a normal distribution estimated on the loss window of the previous plateau - that is when $\mu(l\_window) > plt\_\mu + plt\_\sigma$ where plt_μ and plt_σ are the statistics of the previously detected plateau. This accounts for the continuous fluctuations in the loss function in the online learning and detects when significantly harder samples are observed.

A small buffer with hard samples: in a task-based sequential learning setup, importance weights are estimated on all the training data of the previous task. This is not an option for online learning, as storing all the previous data violates the condition of our setup. On the other hand, using only the most recent sequence of samples would lead to misleading estimates as these few consecutive samples might not be representative and hence do not capture the acquired knowledge correctly. To stabilize the online learning, we use a small buffer of hard samples that is updated at each learning step by keeping the samples with highest loss among the new samples and the current buffer. This is important as previous samples cannot be revisited and hence gives the system the advantage to re-process those hard samples and adjust its parameters towards better predictions in addition to getting a better estimate of the gradient step by averaging over the recent and hard samples. Moreover, the hard buffer represents a better estimate of the acquired knowledge than the few very recent samples and hence allows for a better identification of importance weights.

Accumulating importance weights: as we frequently update the importance weights, simply adding the new estimated importance values to the previous ones as suggested in MAS [1] would lead to very high values and exploding gradients. Instead, we maintain a cumulative moving average of the estimated importance weights. Note, one could deploy a decaying factor that allows replacing old knowledge in the long term. However, in our experiments a cumulative moving average showed more stable results.

After updating the importance weights, the model continues the learning process while penalizing changes to parameters that have been identified as important so far. As such our final learning objective is:

$$\min_{\theta} \mathcal{L}(F(X;\theta), Y) + \mathcal{L}(F(X_B;\theta), Y_B) + \frac{\lambda}{2}\sum_i \Omega_i(\theta_i - \theta_i^*)^2 \quad (5)$$

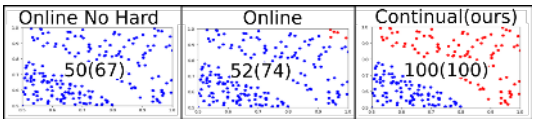


Figure 2: Synthetic experiment: Predictions on first quadrant after training second quadrant. Test accuracy on first quadrant (total test accuracy on both quadrants) overlaid.

where θ^* are the parameters values at the last importance weight update step. Algorithm 1 summarizes the different steps of the proposed continual learning system.

4. Experiments

As a proof-of-concept, we validate our proposed method on a simple synthetic experiment. Later, we evaluate the method on two applications with either weak or self-supervision. First, we learn actor identities from watching soap series. The second application is robot navigation. In both cases, data is streaming and online continual learning is a key factor.

4.1. Synthetic Experiment

We constructed a binary classification problem with points in 4D in/out of the unit sphere. On a sequence of two tasks where each task corresponds to a quadrant, we test the performance of online with no hard buffer, online and our full online continual method. Fig.2 depicts the predictions near the decision boundary in the first quadrant at the end of training on data in the second quadrant. The hard buffer results in better learning (higher total test accuracy), while the full method also avoids forgetting.

4.2. Continual Learning by watching Soap Series

Here, we assume that an intelligent agent is watching episodes from a tv series and learns to differentiate between the faces of the different actors. The agent is equipped with a face detector module that is detecting faces online and a multi-object face tracker. In the case of weak supervision, we assume there is an annotator telling the agent whether two consecutive tracks are of the same identity or not. For the self-supervised case, we use the fact that if two faces are detected in the same image then their tracks must belong to two different actors.

Setup: we start from an AlexNet [15] architecture with the convolutional layers pre-trained on ImageNet [16] and the fully connected layers initialized randomly. The output layer is of size 100. Since the input consists of two tracks of two different identities, we use the triplet margin loss [4] which has been shown to work well in face recognition applications. This has the additional advantage that we don't need to know all the identities beforehand and new actors can be added as more episodes are watched.

Dataset: we use the actor labelling dataset from [3], specifically 6 episodes of The Big Bang Theory (BBT), 4 episodes of Breaking Bad (BB), and one episode of Mad Men (MM)[1]. Note that for BB and MM, the episodes were further split into a total of 22 and 5 chunks, respectively. For each episode we use the frames, detected faces and tracks along with track labels from [3]. Tracks are processed in chronological order, imitating the setting where tracks are extracted in an online fashion while watching the tv series. As a result, the data is clearly non-i.i.d..

For the supervised setup, every tenth/fifth track is held out as test data in BBT/BB respectively as the later has more tracks, 339 tracks BBT compared to 3941 BB. All the other tracks are used for training. As we only have one episode for MM, we decided not to use it for the supervised setup.

For the self-supervised setup, BB turned out to be unsuitable, given that it is an actor centric series with a large majority of the scenes focusing on one actor. To still have results on two series, we do report also on MM in this case, in spite of it being only one episode. Further, the original tracks provided by [3] were quite short (an average of 8/22 faces per track in BBT/MM). Since this is problematic for the self-supervised setting, we use a simple heuristic based on the distance between the faces embedding (based on AlexNet pretrained on ImageNet) to merge adjacent tracks belonging to the same actor.

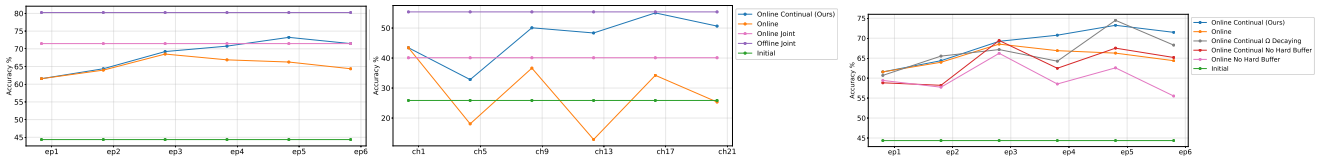
Training: whenever two tracks are encountered belonging to different actors, a training step is performed using the detected faces (one face every 5 frames). If the two tracks contain more than 100 faces, a random sampling step is performed. We use a hard buffer size of 100 triplets and a fixed loss window size of 5. A few gradient steps are performed at each time step (2-3 for the supervised setting, 10-15 for the self-supervised one). We use SGD optimizer with a learning rate of 10^{-4}. Hyperparameters were set based on the first BBT episode, please refer to the supplementary material for more details.

Test: to test the accuracy of the trained model on recognizing the actors in the tv series, we use 5 templates of each actor selected from different episodes. We then compute the Euclidean distance of each test face to the templates, based on the learned representation, and assign the input face to the identity of the template that is closest.

Baselines: to estimate the benefit of our system, *Online Continual*, we compare it against the following baselines:

1. *Initial* : the pretrained model, i.e. before training on any of the episodes.
2. *Online Baseline* : a model trained in the explained online setting but without the MAS importance weight regularizer.
3. *Online Joint Training* : a model trained online, again

[1]Unfortunately, there was an issue with the labels for the other episodes of Mad Men, which prevented us from using these.



(a) (b) (c)

Figure 3: Accuracy on test data at the end of each episode for Big Bang Theory (a) and after chunks 1,5,9,13,17 and 21 of Breaking Bad (b). (c) a study on the importance of the hard buffer and the cumulative Ω average versus a decaying Ω, figure shows the test accuracy after each episode of BBT.

without MAS regularization, but with shuffled tracks across episodes to obtain i.i.d. drawn data.

4. *Offline Joint Training* : a model that differs from Online Joint Training by going multiple epochs over the whole data. This stands as an upper bound.

4.2.1 Weak Supervision Results

Figure 3 (a) shows the actor recognition accuracy evaluated on all the test data of BBT, at the end of each episode. Initially, the Online Baseline (orange) obtains an increase of 20% in accuracy compared to the initial model. Yet it fails to continue accumulating knowledge and improving the accuracy as training continues. After the third episode, the overall accuracy starts to decay, probably because the knowledge learned from these new episodes interferes with what was learned previously. In contrast, our Online Continual Learning system (blue) continues to improve its performance and achieves at the end of the 6 episodes an accuracy that matches the accuracy of the model trained with shuffled data under the i.i.d. condition (Online Joint Training, pink). Offline Joint Training (purple) with multiple revisits to the shuffled data achieves the top performance. Note that this is only 8% higher than our continual learning system trained under the online and changing distribution condition.

Figure 3 (b) shows the accuracy on all the test data of BB, after each 4 chunks while learning the 4 episodes. Clearly this tv series is much harder than BBT. Most of the shots are outdoor and under varied lighting conditions, as also noted in [3]. This corresponds to large distribution changes within and between episodes. Here, the Online Baseline (orange) fails to increase the performance after the first episode. Its accuracy notably fluctuates, probably depending on how (un)related the recently seen data is to the rest of the series. Again, our Online Continual Learning system (blue) succeeds in improving and accumulating knowledge – up to a 100% improvement over the Online Baseline. Like the Online Baseline, its performance drops at times, yet the drops are dampened significantly, allowing the model to keep on learning over time. Surprisingly, it even outperforms the Online Joint Training baseline (pink) and comes close to the Offline Joint training upper bound (purple) that only reaches this accuracy after ten revisits to the training data.

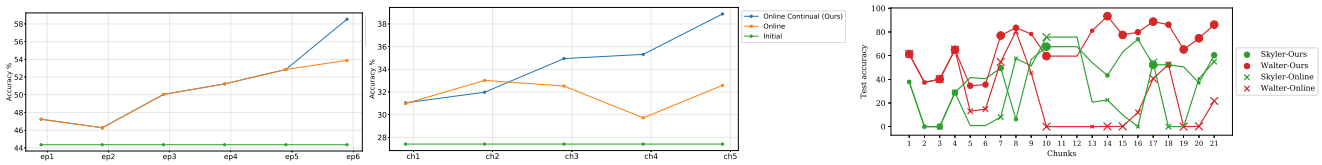
4.2.2 Self Supervision Results

Next we move to the case with self-supervision. This scenario reflects the ideal case where continual learning becomes most interesting. Remember that, as a clue for self-supervision, we use the fact that multiple tracks appearing in the same image should have different identities. We use the six episodes of BBT, although only the first and the sixth episodes actually have a good number of tracks with two persons appearing in one image. Figure 4 (a) shows the accuracy on all the episodes after learning each episode. Note how the Online Learning Baseline (orange) continues to improve slightly as more episodes are watched. It's only when we get to the last episode, with a larger number of useful tracks, that our Continual Online Learning (blue) starts to outperform the Online Learning Baseline.

Figure 4 (b) shows the recognition accuracy on the first episode of Mad Men after each chunk. Similar to the previous experiments our Online Continual learning (blue) succeeds in improving the performance and accumulating the knowledge. We conclude that the ability of continual learning of stabilizing the online learning is clearly shown, both for weak and self-supervised scenarios.

4.2.3 Ablation Study

Next we perform an ablation study to evaluate the impact of two components of our system. The first factor is the hard buffer used for stabilizing the online training and for updating the importance weights. The second factor is the mechanism for accumulating importance weights across updates. In our system we use a cumulative moving average, which gives all the estimated importance weights the same weight. An alternative is to deploy a decaying average. This reduces the impact of old importance weights in favor of the newest ones. To this end, one can set $\Omega_t = (\Omega_{t-1} + \Omega^*)/2$ where Ω^* are the currently estimated importance weights. Figure 3(c) shows the accuracy on all the test data of BBT after each episode achieved by the different variants. The hard buffer clearly improves the performance of both the Online Baseline and Online Continual learning. The buffer



(a) (b) (c)

Figure 4: Self-supervised setting: accuracy on all faces of Big Bang Theory after each episode (a) and of Mad Men after each of the 5 chunks (b). (c) a study on the actors recognition during the course of training, figure shows two main actors test accuracy after each chunk in BB.

Figure 5: Example views in the corridor sequence corresponding to environments A, B, C and D, depicted from left to right.

with hard samples, even if small, gives the learner a chance to re-pass over hard samples and to adjust its gradients for a better estimate of the parameters update step. Additionally, it allows a better estimate of the importance weights used in our Online Continual Learning. The decaying average for the importance weights update, leads more fluctuations due to the higher impact of more recent importance estimates. This allows more forgetting and more bias towards the recent estimate that could be unrepresentative to the overall test data.

Relationship between samples and recognition performance during training: to show how the predictions on the seen actors change over the online training time, we plotted the accuracies per actor after each chunk (for two most frequent characters of BB, to avoid overloading the figure), see Fig.4(c). Marker size indicates the actor's frequency in a chunk; no marker indicates zero appearance. Low frequency in a chunk typically causes the accuracy of the online baseline to drop while our method is more stable.

4.3. Monocular Collision Avoidance

Collision avoidance is the task in which a robot navigates through the environment aimlessly while avoiding obstacles. We train a neural network to perform this task, at test time, based on single RGB images. Training is done with self-supervision where a simple heuristic based on extra sensors, serves as an supervising expert. The deep neural network learns to imitate the expert's control, so cloning its behavior. The task of collision avoidance is best demonstrated in a variety of environments. However, hardware or legal constraints might prevent storing all training data, urging the need for an online learning setup. As the network tends to forget what it has learned over time, the setup is excellent for testing online continual learning.

Architecture: our model takes a 128x128 RGB frame as input and outputs three discrete steering directions. The architecture consists of 2 convolutional and 2 fully-connected layers with ReLU-activations. The training starts with random initialization of the weights and continues with gradient descent on a cross-entropy loss.

Simulation: the experiment is done in a Gazebo simulated environment with the Hector Quadrotor model. The expert is a heuristic reading scans from a Laser Range Finder mounted on the drone and turning towards the direction with the furthest depth. The demonstration of the expert follows a sequence of four different corridors, referred to as A,B,C and D. The environments differ in texture, obstacles and turns, as visible in figure 5.

Training: every 10 steps a backward pass occurs, minimizing the cross-entropy loss, shown in the lower right of figure 6. For each model, three networks are trained with different seeds resulting in the error bars plotted.

Test: the models are evaluated on the entire data sequence as reported in figure 6. The grey bars on the x-axis indicate crossings to new environments. Besides the general online with no continual learning baseline, the performance of following models are given: a scratch initialized model, an online jointly trained model as well as offline. The online joint model has seen all the data once but in an i.i.d. manner. The accuracy of both the online with and without continual learning increases in environments where it is currently learning. Online training without continual training, however, tends to forget the early environments like A, B and C while training in new environments. Especially in environment B and D, the effect is outspoken. The cross-entropy loss in environment D rises for all models, indicating a significant change in the data.

4.4. Proof Of Concept in the Real World

In a final experiment, we apply online continual learning on a turtlebot in a small arena in our lab, see figure 7. The model is on-policy pretrained in a similar simulated environment without continual learning. On-policy refers to the model being in control during training instead of the expert. In the previous experiments, continual learning has proven to be advantageous when big differences occur in the data. In this setup we show that continual learning also provides stabilization during on-policy training within one environment. Again, an expert based on the Laser Range Finder is providing a self-supervisory signal. On-policy learning tends to be more difficult as the data contains a lot of "dummy" samples when the model visits irrelevant states. This data inefficiency causes the model to learn slower and possibly forget along the way. For example, if

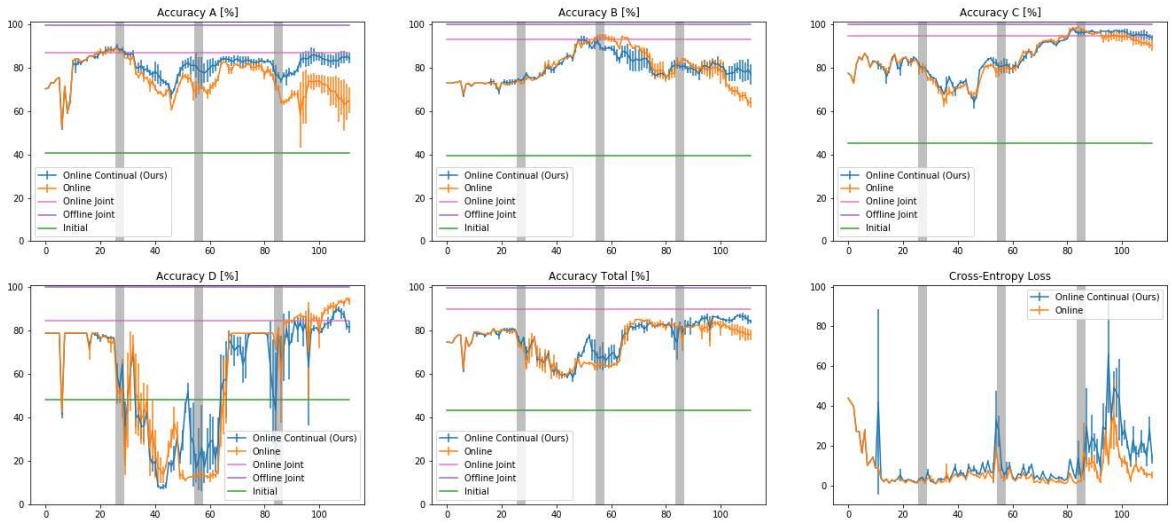


Figure 6: The training accuracy's on the different types of corridors as well as the total accuracy during training on the corridor sequence (A,B,C,D) as depicted in figure 5. Grey lines indicate the transition to a new environment. The lower right figure shows the cross-entropy loss on the recent buffer. The accuracy's of the baselines are added as horizontal lines for the initial model, the jointly trained model both online and offline.

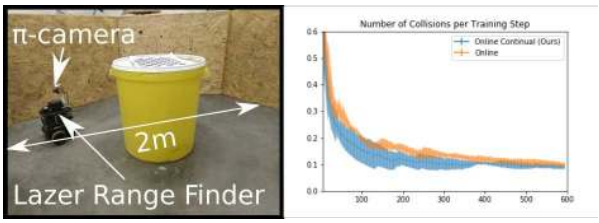


Figure 7: Left: Real-world online and on-policy setup. Right: Number of collisions per training step. Using online continual learning speeds up the training.

the model collides on the left side, the recent data teaches the model to turn right more often. However, after crossing the arena and bumping on the right side, you still want the model to remember its mistakes made earlier. As such, preserving acquired knowledge over time is crucial for on-policy online learning. In figure 7, we show the number of collisions per step over time with error bars taken over three different models. Clearly continual learning helps the model to learn faster, with the number of collisions dropping faster than without it.

5. Discussion and Conclusion

The importance weight regularization appears most effective in online training scenarios when large changes in the learned distribution occur. The closer the online data stream is to i.i.d. samples, the smaller the positive continual learning effect.

In some cases however, continual learning tends to slow down the adaptation to newly seen data. Especially when the new data is much more informative or representative than the old, continual learning initially has a negative effect on the training. In other words, pure online learning is faster to adapt to new changes but therefore also inherently less stable. Ultimately, whether the stabilizing effect of continual learning is advantageous or not, depends on the time scale of the changes in the data.

While in this work we focus on a setting where the network architecture remains fixed, and no new outputs or tasks are added over time, we believe it could also be applied in other settings. For instance in a class-incremental setting, an extra head could be added to the network each time a new category label appears. Alternatively, a projection into an embedding space could be used, as in [7], avoiding the need for a growing network architecture. These are directions for future work.

Due to the limited time, we used data from published datasets in the face recognition experiment allowing quantitative evaluation. However, as future work, we plan to test self-supervised online continual learning on large scale tv-series, thus learning for a longer time.

In conclusion, we pushed the limits of current task-based sequential learning towards online task-free continual learning. We assume an infinite stream of input data, containing changes in the input distribution both gradual and sudden. Our protocol deploys a state of the art importance weight regularization method for online continual learning by detecting when, how and on what data to perform importance weight updates. Its effectiveness is validated successfully for both supervised and self-supervised learning. More specifically, by using our continual learning method, we demonstrate an improvement of stability and performance over the baseline in applications like learning face identities from watching tv-series and robotic collision avoidance.

Acknowledgments: Rahaf Aljundi's PhD is funded by an FWO scholarship. This work was further supported by the CAMETRON research project (GOA) of the KU Leuven and the FWO SBO project Omnidrone.

References

[1] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars. Memory aware synapses: Learning what (not) to forget. *arXiv preprint arXiv:1711.09601*, 2017.

[2] R. Aljundi, P. Chakravarty, and T. Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[3] R. Aljundi, P. Chakravarty, and T. Tuytelaars. Whos that actor? automatic labelling of actors in tv series starting from imdb images. In *Asian Conference on Computer Vision*, pages 467–483. Springer, 2016.

[4] V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. *BMVC*, pages 119.1–119.11, 01 2016.

[5] L. Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998.

[6] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[7] M. Elhoseiny, F. babiloni, R. Aljundi, M. Rohrbach, and T. Tuytelaars. Exploring the challenges towards lifelong fact learning. In *Asian Conference on Computer Vision*, 2018.

[8] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[9] E. Hazan, A. Rakhlin, and P. L. Bartlett. Adaptive online gradient descent. In *Advances in Neural Information Processing Systems*, pages 65–72, 2008.

[10] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. *arXiv preprint arXiv:1710.02298*, 2017.

[11] Y.-C. Hsu, Y.-C. Liu, and Z. Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*, 2018.

[12] T. Hu. Online regression with varying gaussians and non-identical distributions. *Analysis and Applications*, 9(04):395–408, 2011.

[13] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *arXiv preprint arXiv:1612.00796*, 2016.

[14] J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE transactions on signal processing*, 52(8):2165–2176, 2004.

[15] A. Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[17] S.-W. Lee, J.-H. Kim, J.-W. Ha, and B.-T. Zhang. Overcoming catastrophic forgetting by incremental moment matching. *arXiv preprint arXiv:1703.08475*, 2017.

[18] W. Li, J. Huo, Y. Shi, Y. Gao, L. Wang, and J. Luo. Online deep metric learning. *arXiv preprint arXiv:1805.05510*, 2018.

[19] Z. Li and D. Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, pages 614–629. Springer, 2016.

[20] D. Lopez-Paz et al. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6470–6479, 2017.

[21] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*, 24:109–165, 1989.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2014.

[23] A. W. Moore and C. G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13(1):103–130, Oct 1993.

[24] A. Pentina and C. H. Lampert. Lifelong learning with non-i.i.d. tasks. *NIPS*, 2015.

[25] F. Pernici, F. Bartoli, M. Bruni, and A. D. Bimbo. Memory based online learning of deep representations from video streams. *CoRR*, abs/1711.07368, 2017.

[26] F. Pernici and A. Del Bimbo. Unsupervised incremental learning of deep descriptors from video streams. *ICMEW.2017.8026276.*, pages 477–482, 2017.

[27] S. Ramasamy, K. Rajaraman, P. Krishnaswamy, and V. Chandrasekhar. Online deep learning: growing rbm on the fly. *arXiv preprint arXiv:1803.02043*, 2018.

[28] A. Rannen, R. Aljundi, M. B. Blaschko, and T. Tuytelaars. Encoder based lifelong learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1320–1328, 2017.

[29] S.-A. Rebuffi, A. Kolesnikov, and C. H. Lampert. icarl: Incremental classifier and representation learning. *arXiv preprint arXiv:1611.07725*, 2016.

[30] A. Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

[31] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne. Unsupervisedexperience replay for continual learning. *arxiv:1811.11682.*, 2018.

[32] D. Sahoo, Q. Pham, J. Lu, and S. C. Hoi. Online deep learning: Learning deep neural networks on the fly. *arXiv preprint arXiv:1711.03705*, 2017.

[33] S. Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

[34] D. L. Silver, Q. Yang, and L. Li. Lifelong machine learning systems: Beyond learning algorithms. In *AAAI Spring Symposium: Lifelong Machine Learning*, pages 49–55. Citeseer, 2013.

[35] P. Sprechmann, S. M. Jayakumar, J. W. Rae, A. Pritzel, A. P. Badia, B. Uria, O. Vinyals, D. Hassabis, R. Pascanu, and C. Blundell. Memory-based parameter adaptation. *arXiv preprint arXiv:1802.10542*, 2018.

[36] A. L. Strehl and M. L. Littman. Online linear regression and its application to model-based reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1417–1424, 2008.

[37] S. Thrun and T. M. Mitchell. Lifelong robot learning. *Robotics and autonomous systems*, 15(1-2):25–46, 1995.

[38] J. Yoon, E. Yang, J. Lee, and S. J. Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2018.

[39] F. Zenke, B. Poole, and S. Ganguli. Improved multitask learning through synaptic intelligence. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.