Continual Learning Using Task Conditional Neural Networks

Honglin Li, Payam Barnaghi, Senior Member IEEE, Shirin Enshaeifar, Member IEEE, Frieder Ganz.

Abstract—Conventional deep learning models have limited capacity in learning multiple tasks sequentially. The issue of forgetting the previously learned tasks in continual learning is known as catastrophic forgetting or interference. When the input data or the goal of learning change, a continual model will learn and adapt to the new status. However, the model will not remember or recognise any revisits to the previous states. This causes performance reduction and re-training curves in dealing with periodic or irregularly reoccurring changes in the data or goals. The changes in goals or data are referred to as new tasks in a continual learning model. Most of the continual learning methods have a task-known setup in which the task identities are known in advance to the learning model. We propose Task Conditional Neural Networks (TCNN) that does not require to known the reoccurring tasks in advance. We evaluate our model on standard datasets using MNIST and CIFAR10, and also a real-world dataset that we have collected in a remote healthcare monitoring study (i.e. TIHM dataset). The proposed model outperforms the state-of-the-art solutions in continual learning and adapting to new tasks that are not defined in advance.

Index Terms—Adaptive algorithms, continual learning, incremental learning, density estimation.

1 INTRODUCTION

The human brain can adapt and learn new knowledge in response to the changing environments. We can continually learn different tasks while retaining previously learned variations of the same or similar phenomenon and give different reactions under different contexts. Neurophysiology research has found that our neurons are task-independent [1]. Under different context, the neurons are fired selectively with respect to the stimulus. In contrast, most of the machine learning models, in a scalable way, are not capable of adapting to changing environments quickly and automatically using specific neurons corresponding to different tasks. As a consequence, these machine learning models tend to forget the previously learned task after learning a new one. This scenario is known as catastrophic forgetting or interference in machine learning [2].

Catastrophic interference problem in machine learning is one of the inevitable hurdles to implement a general artificial intelligence learning system without constructing a set of models each dedicated to a specific task or different variations and situations in the data [3]. Unable to learn several tasks sequentially, the model must be trained with all the possible scenarios in advance. This requirement is intractable in practice and is different from the lifelong learning goal in continual learning models [4].

Continual machine learning algorithms change over time and adapt their parameters to the changes in data or the learning goal. We refer to the learning goal or a specific part of the data with a learning goal as a task. The learning models are not often equipped with solutions to quickly adapt to the situations which they have seen before if their parameters have significantly changed over time by continual learning. Here we use an example to illustrate the forgetting problem. We train a neural network for two different tasks sequentially. After being trained for each task, the model is represented by parameters \( \theta_0, \theta_1, \theta_2 \) respectively, where the \( \theta_0 \) is the random initialisation weights, \( \theta_1 \) is the weights after learning task 1, \( \theta_2 \) is the weights after learning task 2. We use the linear path analysis [5] to visualise the loss surface. We define \( \theta = \theta_0 + \alpha(\theta_1 - \theta_0) + \beta(\theta_2 - \theta_1) \).

As shown in Figure 1, while the model learns the first task, the parameters change from \( \theta_0 \) to \( \theta_1 \), and the loss for the first task becomes smaller. When the model learns the...
second task, the parameters change from $\theta_1$ to $\theta_2$, and the loss for the first task increases significantly.

A real-world example of this problem is a challenge that we have faced in our remote healthcare monitoring study [6]. We have developed a digital platform and a set of machine learning algorithms to perform risk analysis and provide alerts for early interventions in a use-case scenario to support people affected by dementia. In our user group with the in-home monitoring scenario, the distribution of data is periodical (due to seasonal and environmental effects). In some cases, the data and conditions are and conditions sporadically changed and repeated due to variations in participants’ health conditions. When we use continual and adaptive learning to update the models according to these changes, we face the problem of models not preserving the earlier learned tasks when they reoccur. There are two potential solutions, either maintaining multiple models to respond to different situations or developing models that inherently adapt to the changes and preserve the previously learned tasks as well. Maintaining several models for changing tasks also faces another challenge to detect when a change has occurred and being able to identify if the same or a similar task has previously been observed.

A variety of continual learning methods have been proposed to solve the problems mentioned above. Memory-based approaches replay the trained samples to solve the forgetting problem while learning a new task [7]. Regularisation methods reduce the representational overlap of different tasks [8]–[10]. Dynamic network approaches assign extra neuron resources to new tasks [11]. A more detailed description of these approaches is provided in Section 5. However, the solutions mentioned above assume the models are aware of the task changes in advance, or this information is given manually to the model throughout the learning.

These models cannot detect the changes [8], [9] or utilise the task-specific neurons without knowing the emerging task in advance [12]–[15]. Unfortunately, the task change is rarely known [16] in advance. An ideal incremental learning model must meet the following criteria: 1) The model can learn different tasks sequentially without forgetting the previous ones; 2) The model can obtain the task information and then give different responses to different tasks if required.

We proposed a Task Conditional Neural Network (TCNN), which is a fully automated learning model. TCNN has two main advantages in comparison with the existing solutions: it can detect changes in the data and goals by learning new tasks without forgetting the previous ones; it can give different responses to the different tasks without being informed in advance.

The proposed model overcomes the interference between different tasks by inferring which task the model is encountering with at any given time. TCNN leverages the Probabilistic Neural Networks (PNN) [17] to construct a trainable measure to distinguish different tasks. PNN is a non-parametric model that suffers from significant error buildup when the number of nodes increases. The conventional neural networks also provide a parametric model that suffers from over-confidence in their predictions in dealing with imbalanced and dynamic data. In contrast to the previous models, TCNN can estimate the probabilistic density to find a general representation of the training samples by maximising the task likelihood. In this way, TCNN obtains the task information instead of being informed with the tasks in advance. The schematic description of the TCNN process is shown in Figure 2. During the training step, TCNN optimises the task likelihood measure by maximising the conditional probability of samples given task identity. During the test and run-time, TCNN computes the task likelihood and selects the corresponding neurons to provide a prediction, or it assigns new neuron resources to a new task if the existing resources are not capable of providing a suitable response. TCNN maximise the task likelihood by leveraging a probabilistic layer, shown in Figure 3.

We further combine the replay mechanism with TCNN as Replay Task Conditional Neural Networks (RTCNN) to incrementally learn which samples are out of learning distributions. Generally speaking, while learning the $k$th task, the $i$th task-specific neurons, where $i \in \{1 : k - 1\}$, minimise the task-likelihood of samples from task $k$. By decreasing the impact of the classifier, RTCNN can incrementally learn the out-task samples without decreasing the performance of the $i$th task-specific neurons on task $i$.

Our main contributions in this paper include: i) proposing a novel method to address the catastrophic interference problem; ii) demonstrating how we can incrementally learn the samples from new tasks without decreasing the performance; iii) proposing a model to detect and learn the new tasks automatically. The rest of the paper is organised as follows. Section 2 empirically investigates the cause of catastrophic forgetting problem in machine learning. Section 3 discusses the related work. Section 4 describes the proposed Task Independent Neural Network (TCNN) model. Section 5 demonstrates the experiments and discusses the evaluation results. Section 6 provides an ablation study and Section 7 concludes the paper.

2 The Causes of Forgetting Problem in Continual Learning

Kortge et. al. [18] state that the interference problem in neural networks caused due to the back-propagation rule. This idea is studied by several groups, including Kirkpatrick et. al. and Lee et. al. [8], [9]. They argue that the reason for interference is because the parameter space adapts to new task rapidly and then comprise the previous task but with lower accuracy in responding to the earlier learned task. While learning two tasks sequentially, the model pays attention to the current task. In this case, the parameters change significantly after learning the new task, and if the model is given the earlier task again, it will not respond well until it re-learns it again. French et. al. [19]–[21] argue that the problem is caused due to the overlap in the internal representation of different tasks. This idea is also investigated by Goodfellow et. al. [22], who prove that the dropout [23] can mitigate catastrophic interference. Similarly, Masse et. al. [14] also demonstrate that by deactivating a portion of the neurons before training new tasks, a model can address the catastrophic interference.

Based on the existing studies reported in [12]–[15], the task information is one of the main causes of catastrophic
interference. We investigate different scenarios of informing the model about the changing task information:

$S1$: The task is unknown to the model all the time; $S2$: The model do not need to be informed of the task information in the testing stage [8], [9]. But it cannot detect the changes automatically at the training stage; $S3$: The task information is known at both the training and testing stages [12]–[15]. The model needs to be told which neurons should be activated during the testing stage; $S4$: The model knows what task it is about to perform before the training starts and knows the task changes during the test and run-time [24].

Figure 4 demonstrates how the task information affects the results. The scenarios $S1$ and $S2$ are shown as baseline in Figure 4. In the scenario $S3$, there are many different ways to inform the model about the task identities. Here we use context signal [14], [25] and multi-head approach [12]. The context signal is to add the task identity along with the samples in the input layer. The multi-head is to mask the output layer to make the model only response to the current task. The scenario $S4$ is named as warmup in Figure 4. Warmup allows the model to preserve a small set of samples drawn from the tasks to be learned. Overall, the positive effect of the task information shown in Figure 4 increases in the following order: i) Baseline (no task information), ii) Context signal, iii) Multi-head, iv) Context + warmup, v) Multi-head + warmup.

Since the multi-head approach tells the model in advance which task is about to be performed, the model can determine approximate parameters even the model had not seen the task before. This is why the multi-head approach has a higher overall accuracy after learning the first task.

As shown in Figure 4 a model can address the catastrophic interference problem by using advance information about the changes. The multi-head and warmup explicitly provide all the task information to the model, and this allows them to learn new tasks without forgetting the previous ones. Overall, the more advance information is provided
regarding the task that a model is about to encounter, the more effective model will be in adapting to the new goal or data.

Based on the above example, one can see that the task information is important to address the forgetting problem in continual learning. Informing the task identity can be regarded as maximising the likelihood of \( P(x|T) \), where \( x \) is the samples and \( T \) is the task information. However, to the best of our knowledge, few studies propose a model that can infer the information \( T \) without being told in advance. Li et al. [26] leverage the uncertainty to get the task information at the prediction stage. However, their method cannot detect the changes automatically. Farquhar et al. [27] also suggest that mutual information can be used to identify the changes. However, calculating mutual information can become intractable in large-scale scenarios. The closest work to SI, which is able to detect the changes in the tasks, is by Aljundi et al. [16]. However, Aljundi et al.’s work detects the changes based on the plateaus in the loss surface. In other words, they assume that model keeps learning new tasks continuously without identifying a set of specific and reoccurring tasks. The assumption of continuously learning new tasks increases the complexity of the model in Aljundi et al.’s work and hinders the scalability and applicability of their approach to online and real-world learning scenarios.

In our work, we develop models that can identify tasks and preserve the learned parameters for distinctive tasks. By doing this, the model can quickly respond to a new task based on the previously learned information.

3 RELATED WORK

There are different approaches to address the forgetting problem in continual learning. Parisi et al. [26] categorise these approaches into three groups: Regularisation, Memory Replay and Dynamic Network approaches.

The regularisation approaches find the overlap of the parameter space between different tasks. One of the popular algorithms in this group is Elastic Weight Consolidation (EWC) [9]. EWC avoids significantly changing the parameters that are important to a learned task. It assumes the weights have Gaussian distributions and approximate the posterior distribution of the weights by the Laplace approximation. A similar idea is used in Incremental Moment Matching (IMM) [8]. IMM finds the overlap of the parameter distributions by smoothing the loss surface of the tasks. Zeng et al. [10] address the forgetting problem by allowing the weights to change within the same subspace of the previously learned task. Li et al. [29] address the problem by using the knowledge distillation [30]. They enforce the prediction of the learned tasks to be similar to the new tasks. However, these models require advance knowledge of the training tasks and the task changes.

Memory Replay methods mainly focus on interleaving the trained samples with the new tasks. A pseudo-rehearsal mechanism [31] is proposed to reduce the memory requirement for storing the training samples for each task. In a pseudo-rehearsal, instead of explicitly storing the entire training samples, the training samples of previously learned tasks are drawn from a probabilistic distribution model. Shin et al. [2] propose an architecture consisting of a deep generative model and a task solver. Similarly, Kamra et al. [32] use a variational autoencoder to generate the previously trained samples. However, this group of models is complex to train, and in real-world cases, the sampling methods do not offer an efficient solution for sporadic and rare events. These models also often require advance knowledge of the change occurring.

Dynamic Networks allocate new neurons to new tasks. Yoon et al. [11] propose Dynamic Expandable Networks (DEN) to learn new tasks with new parameters continuously. Similarly, Serra et al. [15] also allocate new parameters to learn new tasks. However, this group of models require the task information to be given to the model explicitly. In other words, the model knows in advance, which neurons should be activated to perform each test task.

4 TASK CONDITIONAL NEURAL NETWORK

The core idea in TCNN is to activate/deactivate the neurons based on the task being processed by the model. Masse et al. [14] and Serra et al. [15] use a similar idea in their work. However, they deactivate the neurons based on knowing the task identities in advance. In other words, their models manually activate or deactivate the neurons corresponding to a task. In TCNN, we select the neurons corresponding to a task by learning and preserving parameters of earlier learned tasks at the training stage and by observing and identifying the task identity at the run-time and processing stage. A probabilistic estimation allows TCNN to determine the task identity by evaluating the model state at any given time of the training and run-time process. The task information determination enables TCNN to learn several new tasks automatically. TCNN measures the (uncertainty of the neural network in processing a task and then uses the previously trained set of neurons and the parameters associated with a task. If the task is a brand new one, it allocates new resources and parameters in combination with the existing ones. Different from the previous uncertainty measure methods [33, 34], TCNN measures the confidence of the neural network without ensembling several neural networks. In other words, the complexity of TCNN to produce confidence is relatively small. Furthermore, the task likelihood in TCNN is trainable in a tractable manner.

4.1 Training Stage

At the training stage, TCNN learns how to process a task. It also obtains the task information by processing the input data. While learning a new task, we maximum the task likelihood of \( P(x|t_k = 1) \), where \( x \) represents the training samples, \( t_k = 1 \) represents the hypothesis of the \( k_{th} \) task being the current one.

TCNN uses a probabilistic layer to estimate the probability density of the input data. The function is shown in Equation (1):

\[
\varphi(x, C) = \exp\left\{ \frac{||g(x) - C||^2_2}{2 \cdot \sigma^2} \right\}
\]

where \( x \) is the training sample, \( g(\cdot) \) represents the joint function of the parameters of the previous layers shown by \( \theta \), \( C \) is the kernel in the probabilistic layer, \( \sigma \) is a hyper-parameter which can be regarded as the radius of \( C \).
Different from conventional fully connected networks, the training parameter in the probabilistic layer is a set of parameters as follows. Assuming there are \( n \) kernels in the probabilistic layer and \( C = \{ c_1, c_2, \ldots, c_n \} \), we measure the confidence by Equation (2) which is inspired by Wedding et al. [35]:

\[
P = \frac{\sum_i^n \varphi(x, c_i)}{\sum_i^n \varphi(x, c_i) + n - \max_i \{\varphi(x, c_i)\} n + n}
\] (2)

where \( P \) is the conditional binary probability of having \( k_{th} \) task given a set of samples and network parameters, and \( P(x|t_k = 1, \theta_k, C_k) + P(x|t_k = 0, \theta_k, C_k) = 1 \), where the \( \theta, C_k \) are the parameters in the task-specific neurons. In the rest of this paper, we use \( P_k \) to represent the probability of the \( k_{th} \) task being the current task that is processed by the model.

The form of the probabilistic layer is also different from the conventional fully connected layers. It is also a parametric layer which makes it different from the layers in PNN as well. Furthermore, combining with the other layers in the network, this layer can estimate the density of the training samples in high dimension by using a limited number of parameters.

The range of Equation (2) is from 0 to 1. It represents the probability that the samples belong to a certain task. Furthermore, the value of \( P \) is significantly affected by the maximum value of \( \varphi(\cdot)_{1:n} \). Because a single task may contain several different classes with different distributions, we would like to find a general representation for all of them. Overall, (2) represents the probability of whether the samples belong to a particular task or not. TCNN maximises the Equation (3) while training and learning for various tasks \( k \) by combining the density estimation with a conventional classification method.

\[
\theta_k^{MLE}, C_k^{MLE} = \arg\max_{\theta_k} P(y|x, \theta_k) + \lambda * \arg\max_{C_k, \theta_k} P(x|t_k = 1, C_k, \theta_k)
\] (3)

The first term is the likelihood as defined in a classification task. The second term is the task likelihood. \( \lambda \) is a hyper-parameter. Let’s remind that \( P(x|t_k = 1, \theta_k, C_k) + P(x|t_k = 0, \theta_k, C_k) = 1 \). Maximising the second term in (3) is equivalent to minimising \( 1 - P_k \) where \( P_k = P(x|t_k = 1, \theta_k, C_k) \), which is easier to implement.

4.2 Prediction Stage

During the test and run-time, TCNN uses task likelihood to decide which neurons should be chosen for the current task or detect whether a change has occurred.

After the model is converged with one task, TCNN decides the confidence interval for that task. The confidence interval is determined by computing the mean and variance of Equation (2) for a set of training samples \( D \). We assume the task likelihood follows a truncated normal distribution within the range from 0 to 1. We define the acceptance area \( P_k \) as Equation (4), where \( \text{CF} \) is the confidence factor define the bounds of interval or acceptance threshold.

\[
\Delta = \text{mean}(P_k) - \text{CF} * \text{std}(P_k)
\] (4)

Assuming TCNN has learned \( K \) tasks, we first calculate the maximum task likelihood of the associated samples to decide which set of neurons to should provide the prediction. We then multiply the task likelihood by the classification result. Let’s assume \( j_{th} \) task-specific neurons produce the maximum likelihood for a given task. The final prediction is given by Equation (5):

\[
\text{Prediction} = TLF \ast f(\theta_j)
\] (5)

Where \( f(\cdot) \) is the corresponding classification function, TLF is a gate function called Task Likelihood Filter (TLF) calculated by Equation (6).

\[
TLF = \frac{\max(0, P_j - \Delta_j)}{P_j - \Delta_j}
\] (6)

Where \( \Delta_j \) is the \( j_{th} \) threshold set by the confidence interval. Since we train the neural network with the mini-batch approach [36], it is more appropriate to calculate the expectation of \( P_j \) during the prediction state.

In summary, if the task likelihood of the test samples fall within at least one confidence interval of the task-specific neurons, TCNN will provide a response based on the current model, or it will raise a change alert instead to indicate that the model is dealing with a new task.

During the prediction stage, there could be some outliers that belong to the learned task. We do not want the model to be susceptible to outliers. We introduce a process to detect the changes, which is shown in Equation (7). At any time range from \( t_a \) to \( t_b \), if \( D(a,b) \geq \eta \), we determine if a change is detected, or otherwise the model will provide a response based on the current structure and parameters. Where \( b - a > 0 \) and \( \eta \) is a pre-defined threshold.

\[
D(a,b) = \sum_{a}^{b} 1 - \frac{TLF}{b-a}
\] (7)

4.3 Replay Task Conditional Neural Network

We use mini-batch [39] to find the general representation of the training samples, TCNN may not be able to produce the confidence properly when there is only a single test sample. One possible solution is to associate several samples to a task and to calculate the expectation of \( \phi(\cdot) \). The performance of this solution is analysed in Section [36]. Another solution is to increase the margin of the confidence between the in-task samples and out-task samples.

To incrementally learn the samples out of training distribution, we combine the replay mechanism with TCNN and introduce Replay Task Conditional Neural Network (RTCNN). RTCNN allows the task-specific neurons to distinguish the in-task and out-task samples efficiently. After learning a task, we store a sub-set of samples associated with that task.. We then re-train the task-specific neurons with all the stored exemplars. For the task-specific neurons for task \( k \), if the exemplar is within the \( k_{th} \) task, the loss function will be as Equation in [3] or we maximise \( P(x|t_k = 0) \) instead. We aim to decrease the task-likelihood of the out-task samples. We do not want the model to change significantly to forget the learned information. The \( \lambda \) parameter is set to a sufficiently large number during the replay process to control the changes in the model. In other words, instead
of learning how to process new tasks, we minimise the task likelihood of samples which are from other tasks.

5 EXPERIMENTS AND EVALUATIONS

We test our model on the Modified National Institute of Standards and Technology (MNIST) handwritten digits dataset. We also use the Canadian Institute For Advanced Research (CIFAR) 10 dataset, which is a collection of images. For a real-world scenario and to address some of the challenges in our healthcare monitoring research, we use the Technology Integrated Health Management (TIHM) dataset. The TIHM dataset consists of several sensor data types collected using in-home monitoring technologies from over 100 homes continuously for six months. The data includes environmental sensory data such as movement, home appliance use, doors open/closed, and physiological data such as body temperature, blood pressure and sleep. The data was fed to a set of analytical algorithms to detect conditions such as hypertension, Urinary Tract Infections and changes in daily activities. A clinical monitoring team used the results of the algorithms on a digital platform that we have developed in our previous work and in some cases verified the results or labelled the false positives. One of the key limitations of our previous work in TIHM was that the algorithms were trained offline, and they did not learn continually. Another limitation was that with using conventional adaptive models, the algorithms changed over time when the environmental or health conditions changed due to seasonal or short-term effects. However, when an earlier learned status is re-observed by the models, the algorithms were not able to perform efficiently due to significant parameter changes. To evaluate the performance of our proposed continual learning and to demonstrate the effectiveness of the model in addressing a real-world problem, we evaluate TCNN on the TIHM dataset and show it can address the challenges mentioned above.

We compare our model with several state-of-the-art approaches in different scenarios as we mentioned in Section 2. Based on the scenarios that are discussed in Section 2, we compare the proposed methods: S1, with the methods in: i) S2: Incremental Moment Matching (IMM) [8], Orthogonal Weight Modification (OWM) [10], Model Adaptation [39] (MA) and Gradient episodic memory (GEM) [40]; ii) S3: Variational Continual Learning [13], Synaptic Intelligence (SI) [12].

For the existing methods, we follow the original settings, as stated in the above-mentioned papers. Since our model has to detect the changes, there will be True Detection (TD) or False Detection (FD). For our model, after learning the task, we feed the batches of test samples of tasks 1 : k + 1 to the model and use Equation (7) to detect the changes. If the model detects the changes that are associated with previously learned tasks 1 : k, the FD increases. If the model detects task k + 1, which is a new task, the TD increases. We also use True Detection Rate (TDR) and False Detection Rate (FDR) to measure the sensitivity of the model. Assuming the time span of the old/new task is n, number of change detected is d and the time interval in Equation (7) is I. TD, FD, TDR, FDR are calculated by Equation (8):

\[ TDR = \frac{\sum TLF}{n} \]

In our experiments, we set the time interval in Equation (7) to 3, the \( \eta = 1 \). For each task, we run the model for 200 time-slots to detect the changes. We assume the data from different tasks arrives in sequence. During the test, the data comes in blocks which contain several samples drawn from the same task. The intuition behind this is that the data coming in a specific time-span may come from a similar distribution. For example, in the healthcare monitoring scenario, a condition that may affect the activity data may last for several days. Consequently, the data from these days will have a similar distribution in the time-span of the short-term condition. In the rest of this paper, the block size represents the number of samples in a block associated with a specific task. While detecting the changes, we set the block to 10. While testing the model, we report the test accuracy with different block sizes. In RTCNN, we store 2000 samples related to each task in MNIST and CIFAR10 datasets, store 45 samples related to each task in the TIHM experiment. Our model has to infer the task information without being informed during the test. If the model fires the right sets of neurons, the test accuracy is the same as training a single model; if the model activates a wrong set of neurons, it will fail to infer the correct task information and the test accuracy will be set to 0. This process is different from the multi-head approach discussed earlier, which may guess the right label even if the model has not come across the current task before. In the following experiments, decision accuracy represents the rate of choosing the correct set of neurons associated with a task. For the proposed methods TCNN and RTCNN, the number in parenthesis represents the block size.

The task likelihood of the experiments can be found in the supplemental document.

5.1 Split MNIST Experiment

The first experiment is split MNIST, which is a benchmark experiment in continual learning field. We split the MNIST to 5 different tasks of consecutive digits. The basic architecture of our model is a multi-layer perceptron with three hidden layers containing 1000, 1000, and 2560 units, respectively. The last hidden layer of the model is connected to a probabilistic layer containing two vectors. The weight factor \( \lambda \) in Equation (3) is set to 2. The factor in Equation (4) is set to 4.

The test accuracy for this experiment is shown in Table 1. When the block size is 1, TCNN cannot distinguish the task information efficiently. However, when the block size is increased to 10, TCNN can infer the task information correctly. In RTCNN, we can obtain the task information efficiently without increasing the block size. The comparison of the test accuracy and decision accuracy is shown in Table 2. The results show that the replay process improves decision accuracy without affecting the test accuracy.
5.2 Split CIFAR10 Experiment

In the second experiment, we test our model on a more complex dataset. We split CIFAR10 to 5 tasks and compare the performance of TCNN with the state-of-the-art methods. The based model is a convolutional neural network contains two convolutional layers and three fully-connected layers. To reduce the number of training parameters, we apply a transfer learning technique [41]. For all the task-specific neurons, they share the convolutional layers, which are associated with a task produced by each task-specific set.

The detection rate are shown in Table 3. The first column represents how many tasks have been learned. Overall, the true detection TD and TDR rates are relatively higher than false detection FD and FDR. Furthermore, if we apply the detection process, the false detection alert FD is relatively small, but the true detection (TD) alert is still sensitive to the changes.

The expectation of the task likelihood of 10 samples to be associated with a task produced by each task-specific neurons is shown in Figure 5. In TCNN, the task-specific neurons have higher task likelihood when the samples come from the corresponding task. However, the margin of the task likelihood between different task-specific neurons is not significant. In RTCNN, the task-specific neurons have larger task likelihood in the case of their corresponding task compared with other sets.

The detection rate are shown in Table 3. The first column represents how many tasks have been learned. Overall, the true detection TD and TDR rates are relatively higher than false detection FD and FDR. Furthermore, if we apply the detection process, the false detection alert FD is relatively small, but the true detection (TD) alert is still sensitive to the changes.

5.3 Healthcare Monitoring Data Experiment

Our last experiment is to evaluate our model on a remote healthcare monitoring dataset. The Technology Integrated Health Management (TIHM) dataset is collected by in-home monitoring sensory devices. As we discussed earlier, the TIHM dataset has been collected from over 100 homes with more than 200 participants in a clinical study aiming to improve the quality of life and to analyse the risk of adverse health conditions in people with dementia. We do not compare our model with the S3 methods. Since there is only one class in each task, providing the task information is equivalent to telling the model which class it is about to perform.

In this experiment, we first evaluate changes in the daily activities of the participants in the study. This data contains three classes: low, medium and high levels of changes in the routine of daily living activities. Compared to the other experiments discussed above, this is a more challenging problem. The TIHM data is unbalanced. The low activity-change class contains 11057 samples; medium activity-change class includes 1146 samples, and high activity class contains only 64 samples. The model should be able to learn several tasks sequentially and also process the unbalanced data automatically. Different levels of activity and their changes also have various characteristics in different participants. In other words, a change in the level of activities to indicate low or medium or high activity does not have the same distribution in all the participants’ data. We split the dataset into training and test sets and then follow the same steps.
as described above for other experiments to learn the three tasks in this experiment. Each task in this experiment only contains one class. The latter means that the test accuracy will be the same as decision accuracy.

The test accuracy is shown in Table 7. Overall, combining the replay mechanism with TCNN improves the accuracy significantly with having a relatively small set of training samples. Since for the S3 methods, the model is known what task to perform in advance and there is only one class in each task, hence the accuracy is 100%.

| Test Accuracy (%) |
|-------------------|
| IMM 33.3          |
| TCNN (1) 78.8     |
| TCNN (10) 98.7    |
| RTCNN (1) 92.7    |

The detection accuracy is shown in Table 8. The higher rate of true detection compared with the false detection shows that the model is sensitive to the changes and confident with the learned data.

| TD | TDR | FD  | FDR  |
|----|-----|-----|------|
| 1 Task | 1.0 | 1.0 | 0.0  | 0.025 |
| 2 Task | 1.0 | 1.0 | 0.0  | 0.0175 |

We also evaluate the applicability of the model in classifying the cases of Urinary Tract Infections (UTIs) in the dataset. UTIs are one of the common causes of hospital admissions in people with dementia. In the TIHM dataset, we have some cases that are tagged by a monitoring team as true positives or false positives. The underlying data associated with these detected conditions are multivariate sensory data coming from sleep, movement, door and physiological monitoring sensors. One of the key limitations in our previous work in this area [38] was that the algorithms had to be trained offline and also they could not adapt to various distributions representing the patient groups that had UTI but with a different manifestation of symptoms. Using TCNN and RTCNN with the TIHM data, the model can incrementally learn different distributions in each class (i.e. positive or negative for UTIs) and to adapt to the changes in the input data. The results of the test accuracy for this experiment are shown in Table 9.

| Test Accuracy (%) |
|-------------------|
| IMM 30.0          |
| TCNN (1) 71.48    |
| TCNN (8) 88.25    |
| RTCNN (1) 78.65   |

6 Discussion

In this section, we analyse the performance of TCNN under unknown task settings. We evaluate how the block size af-
The performance of the model and visualise the density approximated by the probabilistic layer. We also discuss the probabilistic layer and provide an ablation study.

6.1 The performance of TCNN under Task-Unknown Setting

The task-unknown settings represent conditions in which we do not inform the task information to the model at any times. There are two phases in task-unknown settings: i) Training Phase: the model learns a new task; ii) Prediction Phase: the model provides the results with test data and detects the changes. The model will go to the training phase, if and only if a change is detected in the prediction phase. For the analysis purpose, in the prediction phase, after learning task $K$, the model will provide predictions on the test samples of task $K$. After that, the model will give predictions on the test samples of task $K + 1$. In this experiment, we set the block size to 10 and use the split MNIST and CIFAR10 datasets. In the prediction phase, for each task, we test the model on 200 batches. For the visualisation purpose, we use PAA [42] to process the data. The task-likelihood and the average test accuracy on all the tasks are shown in Figure 6.

As shown in Figure 6, while a new task transpires, the task likelihood decreases significantly. The average test accuracy shows that TCNN detects the task changes and adapt to the new tasks quickly without forgetting the previously learned ones. Overall, TCNN provides a unique and novel feature by automatically detecting and adapting to new tasks in a scalable and efficient way.

6.2 Block Size

Although TCNN can detect the task changes, the test accuracy can be affected by the block size. As shown in Table 1 and 4, when the block size is set to 1, the accuracy is low. When the block size is increased, the test accuracy increases significantly. TCNN maximises the task-likelihood by finding a general representation of the training samples. Intuitively providing more samples will enhance determining the task likelihood. As shown in Figure 8, at each time slot, if the block size increases, the performance of TCNN will also increase.

6.3 Analyse the Estimated Density

In this section, we use the split MNIST experiment to visualise the density estimated by the probabilistic layer.

In the split MNIST, we have 5 different tasks. Hence in the probabilistic layer for the $k_{th}$ task, we have two kernels referred to as $C_{k} = \{c_{k1}, c_{k2}\}$. Here we show the joint distribution of $c_{11}$ and $c_{12}$. Figure 9a shows distribution of the two kernels in the probabilistic layer in the first task-specific neurons. Figure 9b shows the sample distribution extracted by the previous hidden layers with function $g_1(\cdot)$.

As shown in these figures, the probabilistic layer estimates the density of the sample distributions successfully.

For comparison, we visualise the sample distribution in the second task-specific neurons. The kernel and sample distributions extracted by the hidden layers with function $g_2(\cdot)$ of the second task-specific neurons is shown in Figure 9c.

In this work, we approximate the distributions that can be classified by the neural network. In other words, we consider the hidden layers as a function $g(\cdot)$, which maps the samples in the training task into specific distributions. We jointly train the classifier and the probabilistic layer to approximate these distributions. As the data changes from in-task (i.e. samples more relevant to a specific task) to out-task (i.e. samples less relevant to a particular task) samples, the function $g(\cdot)$ cannot map the samples to the learned distributions any more. As shown in Figure 9d, the function $g_2(\cdot)$ cannot map the samples from the first task to the kernel distribution trained for task 2 (shown in Figure 9c and Figure 9d).

The difference between the joint distribution of the kernel and the samples is shown in Figure 10. The samples
Fig. 7. Task likelihood with only one sample in the probabilistic layer in the split MNIST experiment. In this case, the task likelihood is not correctly determined. In this example, the second task-specific neurons have a higher likelihood than the first task-specific neurons to the samples from the first task.

are drawn from the first task. The difference between the kernel and sample distribution is small for the first task-specific neurons as shown in Figure 10a, and the difference is significant for the second task-specific neurons as shown in Figure 10b.

6.4 Ablation Study

Without the probabilistic layer to compute the task likelihood, the model cannot decide which task-specific neurons should be activated and will not be able to detect the changes in tasks. We take the 5 task split MNIST to perform an ablation study. As shown in Table 10, the conventional neural network model used in this paper cannot detect changes by the softmax output layer. Farquhar et al. [27] and Li et al. [26] suggest that computing the mutual information by using a Monte Carlo sampling can determine the degree of uncertainty. However, the test process using this approach is computationally extensive and slow. Furthermore, their proposed method needs a large batch of samples to compute a reliable degree for the uncertainty measure. However, Farquhar et al. [27] and Li et al. [26] do not propose a solution to detect a new task with limited samples.

The accuracy of the model without including the probabilistic layer is 0.204. Since we have five tasks in this experiment, the accuracy is the same as a random guess.

We then evaluate how the number of parameters in the probabilistic layer affects the detection accuracy. The number of parameters can be regarded as a general representation of the training task. We want to find the probability density of the samples that can be associated with a specific task. The parameters in the probabilistic layer represent all of these densities. We report the five tasks split MNIST experiment’s detection accuracy with just one parameter in the probabilistic layer. As shown in Table 11, the detection accuracy is decreased dramatically compared with Table 3 which includes the scenario with two parameters in the probabilistic layer.

The task-likelihoods of the first three task-specific neurons are shown in Figure 7. With only one probability density in the probabilistic layer, the model cannot get
The task information correctly. We aim to approximate the distributions which can be classified by the neural network. There are potentially two distributions in each task. The general representation of these two distributions is not sufficient to distinguish different task identities. We visualise the sample distribution from the first task. Then compare it to the kernel distribution of the first and second task-specific neurons. Shown in Figure 11. With only one parameter in the probabilistic layer, the forth task-specific neurons have a smaller difference between the sample distribution and kernel distribution. In other words, the model fails to identify the correct task-specific neurons.

When the number of parameters is larger than the number of classes in the task, the model will no longer be sensitive to the task changes, as shown in Table 12. This is due to the fact that the probabilistic layer overfits the sample distributions.

### 7 CONCLUSIONS

In this paper, we first discuss the reasons for forgetting problem in machine learning when different tasks are given
to a model at different times and demonstrate how providing or acquiring the learning task information is essential to address this issue. We also present a challenge that we have in our healthcare monitoring research and discuss how an automated and scalable model can help to solve this issue in dynamic and changing environments. We then propose a Task Conditional Neural Network (TCNN) model for continual learning of sequential tasks. TCNN is a novel model that provides task-specific neurons corresponding to different tasks.

TCNN can learn and decide which neurons should be chosen and activated under different tasks that are given to a model, without having provided the task information in advance. TCNN can detect the changes in the tasks and learn new tasks automatically. The proposed model implements these features by using a probabilistic layer and measuring the task likelihood given a set of samples associated with a specific task. The proposed model interprets the task likelihood as a binary probability and learns the task likelihoods by utilising a probabilistic neural network.

Our proposed model outperforms the state-of-the-art methods in terms of accuracy and also by detecting the changing automatically. We have also shown how TCNN is used to identify the changes in the data and targets and use the previously learned parameters for each task to detect and predict changes in daily-living activities in our remote healthcare monitoring scenario. The proposed model has a significant impact on creating continual learning methods in dynamic and changing environments in which the data distributions and goals change over time, and the previously learned information is required for a machine learning algorithm when a previously learned state reoccurs.

The future work will focus on measuring the confidence value for a specific task for each neuron. In the current model, a probabilistic layer measures the joint confidence of a group of task-specific neurons. If we can measure the confidence for a single neuron, the efficiency of the model will increase significantly. A solution could be storing the confidence measure for each neuron, given the test samples associated with each task. However, this could also provide several magnitudes of complexity that could make the model intractable. In other words, the confidence of the weights is conditional to the samples instead of having a static value [37]. Identifying a change in the current task can also be improved by a pre-processing decision layer (e.g. a time-series clustering method) or by using a set of ensemble models and an algorithm in a dynamic network to select different subsets of neurons depending on the task.

ACKNOWLEDGMENT

This work is supported by Care Research and Technology Centre at the UK Dementia Research Institute (UK DRI). The work is also partially supported by the EU Horizon 2020 IoT Crawler project under contract number: 779852.

REFERENCES

[1] W. F. Asaad, G. Rainer, and E. K. Miller, “Task-specific neural activity in the primate prefrontal cortex,” Journal of Neurophysiology, vol. 84, no. 1, pp. 451–459, 2000.

[2] M. McCloskey and N. J. Cohen, “Catastrophic interference in connectionist networks: The sequential learning problem,” in Psychology of learning and motivation., Elsevier, vol. 24, pp. 109–165.

[3] S. Legg and M. Hutter, “Universal intelligence: A definition of machine intelligence,” Minds and machines, vol. 17, no. 4, pp. 391–444, 2007.

[4] S. Thrun and T. M. Mitchell, “Lifelong robot learning,” Robotics and autonomous systems, vol. 15, no. 1-2, pp. 25–46, 1995.

[5] I. J. Goodfellow, O. Vinyals, and A. M. Saxe, “Qualitatively characterizing neural network optimization problems,” arXiv preprint arXiv:1411.6544, 2014.

[6] S. Eshkevari, P. Barnaghi, S. Skillman, A. Markides, T. Elsahel, S. T. Acton, R. Nillforooshan, and H. Rostill, “The internet of things for dementia care,” IEEE Internet Computing, vol. 22, no. 1, pp. 8–17, 2018.

[7] H. Shin, J. K. Lee, J. Kim, and J. Kim, “Continual learning with deep generative replay,” in Advances in Neural Information Processing Systems, 2017, pp. 2990–2999.

[8] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, “Overcoming catastrophic forgetting by incremental moment matching,” in Advances in Neural Information Processing Systems, 2017, pp. 4652–4662.

[9] 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,” Proceedings of the national academy of sciences, vol. 114, no. 13, pp. 3521–3526, 2017.

[10] G. Zeng, Y. Chen, B. Cui, and S. Yu, “Continuous learning of context-dependent processing in neural networks,” arXiv preprint arXiv:1810.01256, 2018.

[11] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, “Lifelong learning with dynamically expandable networks,” arXiv preprint arXiv:1708.01547, 2017.

[12] F. Zenke, B. Poole, and S. Ganguli, “Continual learning through synaptic intelligence,” in Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017, pp. 3987–3995.

[13] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, “Variational continual learning,” arXiv preprint arXiv:1710.10628, 2017.

[14] N. Y. Massé, G. D. Grant, and D. J. Freedman, “Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization,” Proceedings of the National Academy of Sciences, vol. 115, no. 44, pp. E10467–E10475, 2018.

[15] J. Serrà, D. Surís, M. Miron, and A. Karatzoglou, “Overcoming catastrophic forgetting with hard attention to the task,” arXiv preprint arXiv:1801.01423, 2018.

[16] R. Aljundi, K. Kelchtermans, and T. Tuytelaars, “Task-free continual learning,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 11254–11263.

[17] D. F. Specht, “Probabilistic neural networks,” Neural networks, vol. 3, no. 1, pp. 109–118, 1990.

[18] C. S. Kortge, “Episodic memory in connectionist networks,” 1990.

[19] R. M. French, “Using semi-distributed representations to overcome catastrophic forgetting in connectionist networks,” in Proceedings of the 13th annual cognitive science society conference, 1991, pp. 172–178.

[20] ——, “Semi-distributed representations and catastrophic forgetting in connectionist networks,” Connection Science, vol. 4, no. 3-4, pp. 365–377, 1992.

[21] ——, “ Catastrophic forgetting in connectionist networks,” Trends in cognitive sciences, vol. 3, no. 4, pp. 128–135, 1999.

[22] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, “An empirical investigation of catastrophic forgetting in gradient-based neural networks,” arXiv preprint arXiv:1312.6213, 2013.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” The journal of Machine Learning Research, vol. 15, no. 1, pp. 1929–1958, 2014.

[24] M. K. Hetherington, “Catastrophic interference is eliminated in pretrained networks,” in Proceedings of the 15th Annual Conference of the Cognitive Science Society, 1993, pp. 723–728.

[25] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” arXiv preprint arXiv:1411.1784, 2014.

[26] H. Li, P. Barnaghi, S. Eshkevari, and F. Ganz, “Continual learning using bayesian neural networks,” 2019.
[27] S. Farquhar and Y. Gal, “Towards robust evaluations of continual learning,” arXiv preprint arXiv:1805.09733, 2018.

[28] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, “Continual lifelong learning with neural networks: A review,” Neural Networks, 2019.

[29] Z. Li and D. Hoiem, “Learning without forgetting,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 12, pp. 2935–2947, 2018.

[30] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” arXiv preprint arXiv:1503.02531, 2015.

[31] A. Robins, “Catastrophic forgetting, rehearsal and pseudorehearsal,” Connection Science, vol. 7, no. 2, pp. 123–146, 1995.

[32] N. Kamra, U. Gupta, and Y. Liu, “Deep generative dual memory network for continual learning,” arXiv preprint arXiv:1710.10368, 2017.

[33] T. Pearce, M. Zaki, A. Brintrup, and A. Neel, “Uncertainty in neural networks: Bayesian ensembling,” arXiv preprint arXiv:1810.05546, 2018.

[34] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” in international conference on machine learning, 2016, pp. 1050–1059.

[35] D. K. Wedding II and K. J. Cios, “Time series forecasting by combining rbf networks, certainty factors, and the box-jenkins model,” Neurocomputing, vol. 10, no. 2, pp. 149–168, 1996.

[36] Honglin Li is currently pursuing his Ph.D. in machine learning at the Centre for Vision, Speech and Signal Processing (CVSSP) at the university of Surrey. His research is mainly focused on continual and incremental learning in non-stationary environments.

Payam Barnaghi is Professor of Machine Intelligence at the Department of Electronic and Electrical Engineering and a member of the Centre for Vision, Speech and Signal Processing (CVSSP) at the University of Surrey. He is technical lead of the Department of Health/NHS TIHM for Dementia project and Healthy Home programme lead and Deputy Director of Care Research and Technology Centre at the UK Dementia Research Institute (UK DRI). His research interests include machine learning, Internet of Things, semantic computing and adaptive algorithms and their applications in healthcare. He is a senior member of IEEE and a Fellow of the Higher Education Academy.

Shirin Enshaeifar is a Lecturer in Machine Learning and Deputy Technical Lead of the Technology Integrated Health Management (TIHM) project in the Department of Electronic and Electrical Engineering at the University of Surrey. Her research interests include biomedical signal processing, machine learning and data analysis with application in brain-computer interface (BCI), technology enabled healthcare services, and Internet of Things (IoT).

Frieder Ganz is a Senior Scientist at Adobe focusing on machine learning and computer vision tasks within the domain of digital imaging. He received his PhD from the University of Surrey. His research interests include machine learning and how human and machine perception can become closer.