Task Detection in Continual Learning via Familiarity Autoencoders

Maxwell J. Jacobson, Case Q. Wright, Nan Jiang, Gustavo Rodriguez-Rivera, and Yexiang Xue
Department of Computer Science, Purdue University
{jacobs57, wrigh404, jiang631, grr, yexiang}@purdue.edu

***Abstract*—Continual learning requires the ability to reliably transfer previously learned knowledge to new tasks without disrupting established competencies. Methods such as Progressive Neural Network [1] accomplish high-quality transfer learning while nullifying the insidious problem of catastrophic forgetting. However, most module-based continual learning systems require task labels during operation – a constraint that limits their application in many real-world conditions where task indicators are opaque. This paper proposes a task detector neural algorithm to acquire task information while maintaining immunity to forgetting. Our proposed task detector allows progressive neural networks (and many similar systems) to operate without task labels during test time. Our task detector is built from familiarity autoencoders which recognize the nature of the required task from input data. We demonstrate the generality and effectiveness of our approach through experiments in video game playing and automated image repair. Our results show near-perfect task recognition in all domains ($> .99$ F1), rewards above published single-task scores in MinAtar, and realistic image repairs on damaged human face pictures. The performance of our integrated method is nearly identical to the progressive systems equipped with ground-truth task labels[1].**

I. Introduction

Intelligent systems benefit greatly from the ability to learn new tasks on-the-fly. For example, a game playing agent with the option to add new games to its skillset without re-training would be far more compelling than one without. Extensibility of this sort is possible through the open problem of continual learning — the ability to learn from and adapt to the environment in a continuous fashion [2], [3]. The key towards continual learning is the ability to *transfer* knowledge learned from one task to new tasks, with little or no fine-tuning. Moreover, continual learning needs to overcome *catastrophic forgetting* [4], [5], a disrupting effect that neural networks lose previously learned knowledge once trained on a new task.

Progressive neural systems are a recent forward step towards continual learning. Research along this line initiated from the Progressive Neural Network [1] and has been followed by Dynamically Expandable Networks [6], Improvised Progressive Neural Network [7], PathNet [8], and a host of other architectures. Progressive systems are a class of neural network continual learners that dynamically grow new subnetworks to accommodate new tasks. In progressive systems, old parameters are frozen or otherwise controlled to prevent catastrophic forgetting, and new parameters are transfer learned from these old parameters without changing them. As an example, the upper left panel of Fig. 1 shows a progressive neural network, where new columns of neural net layers are added for each new task, and lateral connections from previous tasks are introduced to enable transfer learning.

Despite its great success, progressive neural systems (and other continual learning approaches) require task labels to work properly. This dependence is notable in Fig. 1 during the output selection operation. Without a correct task label during testing, the progressive net cannot know which subnetwork to choose in solving a task. Other architectures capable of continual learning such as mixture-of-experts (MoE) [9] show the same property. This is a significant limitation towards continual learning – in real-world scenarios, task labels are often unavailable, unclear, or expensive to collect. More often, the learning agent has to quickly identify the task to make an accurate prediction.

Consider the example of a game playing robot. At any time, the robot may be asked to enter a new training phase to learn a new game. The robot is expected to continual learn from previous games to the next game. A progressive system that already performed well on games A, B, and C, could learn D — training only on the new data from D while extracting knowledge from networks associated with A, B, and C. However, when playing a game after training, the system would need to be told whether it was engaging with game A, B, C, or D. It could not infer this information without the risk of catastrophic forgetting.

The purpose of this paper is to introduce a method of automatic task detection for continual learning systems, thereby increasing their test-time autonomy. The primary challenge here is to overcome catastrophic forgetting. A naive idea would be the addition of a classifier network to predict the task label from the input data. This approach is flawed because the classifier itself would be susceptible to catastrophic forgetting as it is not built progressively. While it could be retrained on each new task, this would require a full retraining involving both the current and all previous tasks. Additionally, the training data from previous tasks would need to be accessible for all new tasks – which may not be possible when dealing with massive or volatile datasets.

Our task detector harnesses a growing array of Familiarity AutoEncoders (FAEs) to discover the task label and is not susceptible to any amount of catastrophic forgetting. The high-level idea of the task detector is shown in the lower panel of Fig. 1, where we add one FAE for each new task. The FAE is trained to reconstruct input data from the task it is associated with. This process is conducted simultaneously alongside the progressive system. For example, in a continual learning setting where the AI agent must learn to play a variety of video games, each familiarity autoencoder would be trained to reconstruct frames from the game they are

[1]Code is available at: https://github.com/arcosin/Task_Detector

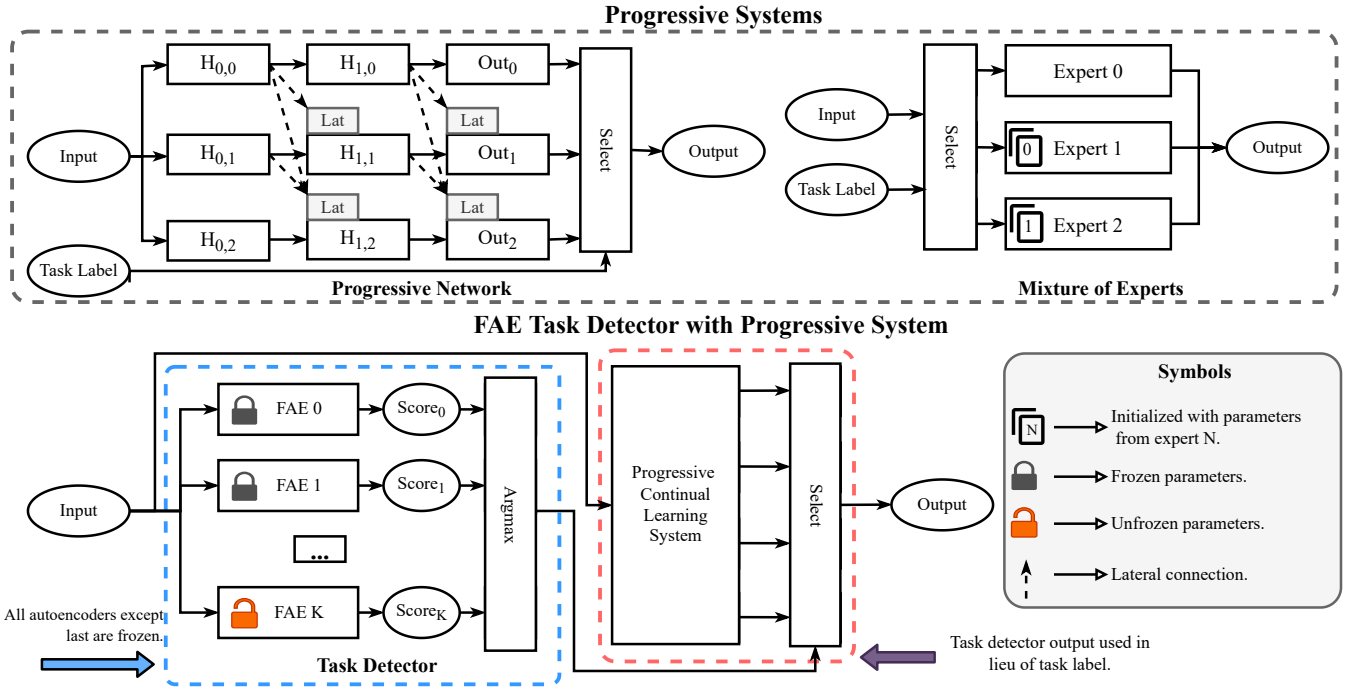

Fig. 1: Framework of our proposed task detector and progressive system. **(Top Left)** A 3-column progressive network (with rows and columns transposed for clarity). Each row network is trained on one task and the input is fed to all three rows. Intermediate representations are passed through lateral connections (dashed lines) to allow transfer learning between tasks. Lateral connections let layers "mix-&-match" representations from previous layers across new tasks. The final output is selected using the task label — a vital function that is sometimes non-trivial when the label is difficult to acquire. **(Top Right)** A mixture-of-experts (MoE) using finetuned copies to grow progressively. This scheme has weaker transfer learning than the progressive net, but slower parameter growth and more flexibility for things like optional re-training. **(Bottom)** Our task detector architecture when added on to a progressive system such as the progressive network or an MoE. The task detector runs each FAE to generate a recognition score — the $\arg\max$ of which is a prediction of the task label and can be used to select the desired sub-network in downstream progressive systems. Because the FAEs are independently and sequentially trained, no catastrophic forgetting occurs. Adding a new task only requires training a new FAE for that task.

assigned. During test-time, the input data of a task is sent to all of the FAEs. The task label prediction is then the index of the FAE which produces the best reconstruction. Our task detector is not susceptible to catastrophic forgetting because each FAE is trained independently with only the data from its assigned task. Besides vanilla autoencoders, this paper also implements FAEs with variational autoencoders (VAEs) and adversarial autoencoders (AAEs). We found that both lead to more stable task predictions in noisy environments.

In the game-playing robot scenario, the robot can expect task labels during a training phase (as it is only training on one game at a time) but it cannot expect them after that – it must infer the game it is playing based on sensor input. Our FAE task detector would allow the robot to do precisely this without storing old data or risking catastrophic forgetting.

We demonstrate the efficacy of the task detector in the Atari, MinAtar, and CelebA Image Repair domains. Our experimental results show that:

- Our task detector leads to **accurate task predictions**. In each domain, it performs a near-perfect classification ($> 99\%$ accuracy / F1 score) — a result that is maintained even when inputs are injected with unexpected test-time noise. In contrast, a simple classifier invariably deteriorates from catastrophic forgetting and yields much lower scores.
- Our task detector **combines well with downstream tasks** when integrated into a progressive system. Applying our method to the domain of automatic image repair, the error over 7 restorations are nearly identical to those made from progressive systems equipped with ground-truth task labels. When applying our method to a set of 5 MinAtar games, we again show nearly identical results compared to an optimal progressive neural network with access to ground-truth task labels. The combined system in both experiments outperforms other continual learning methods including elastic weight consolidation and task replay.

II. Preliminaries

A. Continual Learning & Progressive Systems

Continual learning is a high-level goal in the journey toward general artificial intelligence [10]. Essentially, a continual learning system is capable of constantly improving while remaining stable on previous tasks. Significant work has taken place examining machine learning in evolving environments across many algorithms and settings [11]–[13]. Variables like task length, task repetition, whether task transitions are gradual or discrete, and whether tasks themselves are discrete or continuous can greatly affect the method required to learn a domain of tasks. In addition, the

type of machine learning applied to the task (unsupervised, supervised, reinforcement, etc), the complexity of the task, and the requirements for remembering old knowledge are also important factors.

Related to continual learning, transfer learning is the process of transferring competence from one machine learning system learning a task to another system learning a related task [14]. It is deeply entangled with the concept of continual learning in that a powerful method of transfer learning would be a possible basis toward a full continual learning system. Many solutions to transfer learning exist, the simplest being finetuning, in which a trained model is retrained on a new task, often with additional parameters. While finetuning is suitable for basic transfers, it manifests behaviors that make it unsuitable for continual learning. Chiefly, a deleterious effect known as *catastrophic forgetting* [4], [5], which occurs when important parameters within the network are changed to fit the new data, compromising the network's ability to handle previously-learned operations.

A summary of modern continual learning can be found in [15]–[17]. The most popular overarching methods for continual learning include modulation of gradients [18]–[21], knowledge distillation/replay [22]–[24], and modularity-based approaches [1], [6]–[8], [25], [26]. The last class of approaches is particularly relevant to progressive systems – a class of neural network continual learners that dynamically grows new sub-networks to accommodate new tasks. Old parameters are frozen or otherwise controlled to prevent catastrophic forgetting, and new parameters are transfer learned from these old parameters without changing them.

B. Progressive Neural Networks

The first progressive system was introduced in [1], describing progressive transfer learning in neural networks through the creation of connected sub-networks. Unlike conventional finetuning which does transfer learning only at initialization, progressive nets keep a collection of sub-networks — often referred to as columns — which share information through lateral connection. These lateral connections are layers belonging to a column and taking as input the hidden representations from previous columns. Lateral connections allow information computed in the current column and all previous columns to be combined in service to the current task, without disrupting the parameters in the previous columns. To summarize the full algorithm in terms of continual learning, progressive neural networks maintain one column per task with lateral layers connecting them. As more tasks are added, old columns — along with their incoming laterals — are frozen to maintain their parameters, with transfer learning occurring only through new laterals. Refer to the upper left panel of Fig. 1 for a closer look at the progressive network architecture.

While progressive neural networks and other progressive systems sometimes underperform compared to other methods, they are among the most effective when true forgetting immunity must be guaranteed. Progressive nets prove to be a powerful model for many task sets. In *Progressive Neural Networks*, experiments demonstrate that the system performs well in various reinforcement learning tasks including maze, Atari, and pong variant games. Further, [27], [28] demonstrate the breadth of application for this form of transfer learning. However, progressive networks are not without limitations – for example, the rate of parameter growth and the inefficient information density of new parameters. Another limitation is the necessity of task labels, even after training is complete.

The necessity of task labels in progressive neural nets originates from the structure of the interconnected columns in the network. When the model forward propagates, it runs the input through all columns laterally connected to the task column — allowing finetuned representations of the input to be accessed by new columns of the network. Though many columns may be executed, only the column associated with the current task generates useful output. The important job of task identification is then to select which column actually represents the function associated with the current task. This is a property shared by many progressive systems and continual learners more generally.

C. Autoencoders & Novelty Detection

Autoencoders [29] are a robust class of architectures operating under the premise of restricting the bandwidth of the information to a small latent vector, thus assimilating abstract features of the learned data. One important use case of autoencoders is as an anomaly detection tool. Anomaly detection — also sometimes called novelty detection — is the task of identifying anomalies, outliers, or unusual data records among a class [30]. This task is similar to one-class classification, in which ML systems must decide whether a data record is part of a single class or not. Anomaly detection autoencoders [31] accomplish this through learning reconstruction. An autoencoder trained to reconstruct data from a certain class will perform poorly in reconstructing data sufficiently different from what it knows. The poor performance can be measured by a novelty criterion — usually an error function like mean squared error or cross-entropy loss. After selecting an upper bound and lower bound of error, the autoencoder will be able to determine the unique status of a data record by yielding an error value outside the selected threshold. As autoencoder novelty detectors intrinsically learn an approximation of the input's distribution instead of a hard boundary, they do not require any examples of anomalies during training. Notably, this novelty detection strategy can be easily reversed. By negating the novelty criterion, a measure of familiarity detection can be found, with higher scores corresponding to a greater likelihood that tested data is similar to previously learned training data.

III. Methodology

A. Motivations

Without the ability to recognize tasks from input data, many continual learning systems are forced to rely on task labels, which are not always feasible to obtain in real-world applications. However, predicting a task label from input data is more difficult than it may appear. A naive approach may be the implementation of a simple classifier with each class corresponding to a task label. This approach is flawed because the classifier itself would be susceptible to catastrophic forgetting whenever new classes are added. While it could be retrained on each new task, this would

require a full retraining involving both the current and all previous tasks. All of that data would need to be continually accessible for all new tasks, something that may be difficult when dealing with massive or volatile datasets. Another option is an array of probabilistic binary classifiers indexed to each task. The probability of the positive class calculated by each classifier would act as a recognition for that classifier's task. This solution is better, but still insufficient as no negative examples would be available from future tasks. This elucidates an important point: the key to good continual task detection is not to "draw a line" separating the appearance of each task, but to measure the *essential closeness* of an input to previously seen inputs of the same task.

Assumptions. Our solution to task recognition relies on one important assumption: the task must be recognizable based only on input data from that task. Learning two tasks that require different outputs but receive the same input would break this assumption. Many situations like this can be circumvented with a more appropriate input representation — for example, two video games that appear the same but exhibit different mechanics can be represented recursively or as a list of trajectories instead of as single frames.

B. Task Detector

Big Picture. Our method uses an array of small neural nets — one for each task — that each produces a score for a given input. The score is a measure of how close the data record is to previously seen data from the indexed task. For our prediction, we select the task indexed to the neural net with the highest reward.

Anomaly detection autoencoders produce a positive score representing how novel a piece of data is compared to previously learned data. By negating this score, we can ascertain a quantification of how familiar a piece of data is instead. Essentially, by simply negating the novelty criterion, we can create a familiarity autoencoder (FAE).

Consider a task detector containing an array of FAEs and a progressive neural network for task learning. Upon initialization, the task detector is composed of just one FAE indexed to a progressive network's first column. The initial FAE can be trained on the same data as the progressive network column — though a separate pre-processing step may be necessary to achieve optimal results. The FAEs are trained using self-supervised learning to deconstruct and reconstruct inputs through the latent vector. Once the column network has finished training, its parameters are frozen, as are the parameters of the FAE.

As new tasks are learned, more FAEs are added alongside progressive net columns. Each FAE remains indexed to just one column net, and is only trained on data from the task associated with that column. Immunity to catastrophic forgetting is preserved because the FAEs are only trained on data from a single task.

During test time, the FAEs begin their familiarity measurement role. So long as a sufficiently low-capacity latent vector has been selected, the network will do one of two things. If the unlabeled input does not belong to the task associated with the FAE it will reconstruct the data incorrectly — either transforming it into a false version of itself more similar to the associated task, or transforming it into incomprehensible garbage data. However, if the input does belong to the associated task, the FAE will reconstruct it more effectively. Therefore, the detector with the smallest reconstruction error is most likely to be indexed to the correct task. The familiarity score from each FAE can be calculated as the negative error — defined by the novelty criterion — between the input and the reconstruction. Some candidates for this function include mean squared error, mean absolute error, and cross-entropy. Finally, the predicted column is selected as the $\arg\max$ of all the negative novelty scores. The task detector classification can be formally defined as:

$$\hat{k} = \arg\max_{i \in \{0,\ldots,K\}} - \zeta(x, d_i(x)) \tag{1}$$

where $\hat{k}$ is the predicted task label, ζ is a novelty criterion — a simple measure of error between inputs (e.g. MSE, MAE, etc), and d_i is the FAE at index i.

Similar task detection schemes using autoencoders for recognition have been implemented in other concurrently developed works [32], [33]. However, their application domains were not in sequential continual learning and had major differences in their training and testing procedures. Our task detector is the first sequential continual learning module that is immune to forgetting and allows intrinsic task awareness by integrating familiarity detectors into a progressive system.

Variants. A useful variant of the task detector is one implemented with a variational autoencoder (VAE) [34] or an adversarial autoencoder (AAE) [35]. By modeling the distribution of inputs within the latent space and factoring this into the network's objective as a sort of divergence loss, VAE possesses limited generative capabilities. As a familiarity detector, a VAE would be capable of reconstruction with much finer detail, distinguishing between tasks that are similar but for a few important distinctions. AAE also have generative capabilities, but in these networks they stem from the use of a discriminator network similar to those used in GANs [36]. Finally, in cases where tasks produce different score distributions, a normalization step can be added to standardize the score range, though this was not found to be necessary in our experiments.

C. Limitations & Mitigation Techniques

The primary limitation of our task detector is its memory-intensive nature. Alongside an already cumbersome progressive network, the whole system is likely to have a much greater memory footprint than is necessary to solve the tasks. This parameter overhead can also increase training time. Thankfully, mitigation of this issue is straightforward: the FAEs in the detector are natural candidates for compression and trimming. As they are fully distinct networks, compression can be carried out in a separate process even as new tasks are learned. Even if compression leads to a reduction in reconstruction quality, it is unlikely to cause enough damage to change the final classification. Furthermore, unused FAEs can be expelled to less expensive disk space to free up memory until they are needed again.

IV. Experiments

Our FAE-based task detector was evaluated on four experiments. In the MinAtar experiment and the image repair experiments, we implemented full progressive systems

	Avg. F1	Asterix Reward	Breakout Reward	Freeway Reward	Seaquest Reward	Space Invader Reward	Data usage per task (# experiences)
With Task Labels							
Prognet + Task labels	*N/A*	24.7	18.3	35.7	10.6	11.8	0
Without Task Labels							
Prognet + Task Detector (ours)	**99.3**	**26.1**	**17.0**	**32.6**	**10.9**	11.6	**0**
Task Replay [3]	*N/A*	0.6	0.6	0.2	0.0	**26.7**	100000
EWC [20]	*N/A*	*	*	*	*	*	256
Without Continual Learning							
Published DQN [37]	*N/A*	~ 13	~ 8	~ 48	~ 15	~ 40	*N/A*

(a) **MinAtar Game Playing**. Recognition F1 and Average Reward over 300 Episodes in 5 MinAtar Games. F1 score shows the quality of task prediction, while episode reward shows the quality of the whole agent in playing the games. Data storage shows how many experiences need to be stored between tasks. Our progressive DQN + task detector performed very well in the MinAtar games, even beating some of the published results. These results show that the progressive system maintained competency across all games while task replay resulted in weaker transfer and less forgetting resistance. EWC failed to converge. Other methods also required extra experiences to be stored across tasks, while our method did not.

	Recognition Accuracy	Denoise MSE	Colorize MSE	Inpaint MSE	PF MSE	V2H MSE	I2V MSE	E2DF MSE	Total MSE
With Task Labels									
Prognet [1]	*N/A*	**99.1**	**113.5**	55.7	92.3	105.6	111.0	11.5	**588.7**
MoE [9]	*N/A*	242.7	226.4	**38.0**	**85.8**	**31.1**	**52.0**	**9.2**	685.2
Without Task Labels									
Prognet + Task Detector (ours)	**99.6**	**99.1**	**113.7**	55.5	91.9	105.4	111.0	11.4	**588.0**
Prognet + Classifier	14.3	241.4	261.5	157.0	169.3	214.5	520.7	11.5	1575.9
MoE + Task Detector (ours)	**99.6**	242.6	226.4	**38.0**	**86.9**	**31.3**	**52.2**	**9.2**	686.6
EWC [20]	*N/A*	336.2	338.0	314.0	271.0	326.4	356.1	9.8.	1951.5

(b) **Image Repair.** Recognition Accuracy and Total Mean Square Error (MSE) across 7 image Repair Tasks. Overall accuracy and total MSE over the whole testing set (900 images) for each task are shown. The task detector's excellent classification accuracy alongside the progressive neural net and the mixture-of-experts translated to a similar average error measurement when compared with ground-truth. Both of these progressive systems outperformed elastic weight consolidation. The baseline classifier is only tested with the prognet as it is sufficient to show the effects of catastrophic forgetting on the upstream classifier.

	Accuracy	Averaged F1-Score
Task Detector (ours)	**99.96**	**99.96**
Classifier	77.58	70.23

(c) **Atari Game Recognition**. Recognition across 9 Atari Games. The Task Detector achieves near perfect classification **(99.96% F1 score)**, which attains more than 20% improvement than the baseline. This is compared to the baseline classifier with no catastrophic forgetting mitigation.

	Accuracy	Averaged F1-Score
Task Detector (ours)	89.44	86.31
VAE Task Detector	**99.89**	**99.89**
AAE Task Detector	93.19	93.56
Classifier	37.67	25.39

(d) **Noisy Atari Recognition**. Atari frame classification with test-time Gaussian noise infusion. The standard autoencoder FAE is compared with a variational autoencoder variant and an adversarial autoencoder variant. The Task Detector displays remarkable noise resistance, especially when implemented with VAEs — the results are still nearly perfect even with the added noise.

TABLE I: Results for all experiments. Methods including "+ Task Labels" are supplied with ground-truth task labels, while all others operate only on input data. Across all experiments, our task detector nearly matches the performance of models with ground-truth task labels and – when matched with a downstream progressive system – outperforms all baselines.

integrating our task detector to play arcade-style games and operate on damaged images respectively. In the Atari and noisy Atari experiments, we exclusively implement the task detector. For all experiments, sub-tasks were trained sequentially with no direct access to data from previous tasks. These *training sessions* involved training the task model and the task detector on the same data but independently. All code for this paper was written in Pytorch [38] and used the Doric library[2] for implementing progressive neural networks.

A. MinAtar Games

Setup. MinAtar [37] is a pixelated and minimalized version of classic Atari games. Five games were available: Asterix, Breakout, Freeway, Seaquest, and Space Invaders[3]. Each game frame is represented as a $10 \times 10 \times 10$ matrix with different game elements split into 10 channels. All game elements are single pixels. The player in each game has 6 possible actions available.

A progressive network was used to implement the Q-network and target network in a DQN [39] — the code for this being directly adapted from the exemplar MinAtar DQN to ensure compatibility with other works. For comparison, two other agents were also trained on each game: a replay agent that preserved its buffer across games and ensured equal representation in the buffer at the start of each training session, and an elastic weight consolidation (EWC) DQN

[2]https://github.com/arcosin/Doric

[3]https://github.com/kenjyoung/MinAtar

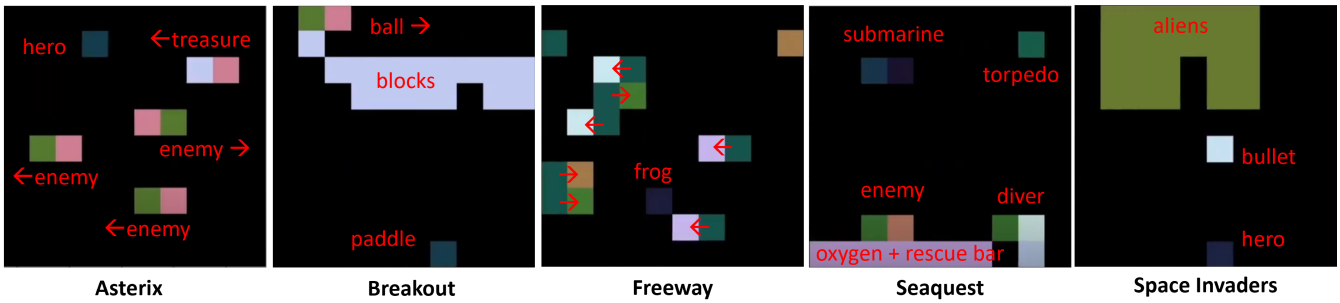

Fig. 2: Example frames from the MinAtar experiment. Our method was able to distinguish frames and respond in real-time with better results than with task replay. This means frames can even be randomly interlaced between games during testing with the same results as clearly separated games. Results can also be viewed here: https://youtu.be/YRkzThwXRSI.

with penalty updated after each training session. The network architecture was a single ReLU-activated convolutional layer (16 channels, 3×3 kernel, stride 1), followed by a ReLU-activated fully connected section with 128 neurons. The progressive network version of the task model was the same, except with the addition of lateral connections mirroring the architecture of their parent layers. Each FAE included one ReLU-activated convolutional layer (8 channels, 3×3 kernel, stride 1), followed by two ReLU-activated fully connected layers with 32 then 800 neurons, and then a transpose convolutional layer (10 channels, 3×3 kernel, stride 1, padding 1). Mean squared error was used as a training criterion and for familiarity detection. While the expanding nature of the algorithm can be memory intensive, this experiment required only 170KB of parameters for each FAE.

Results. Our FAE-based task detector combined with a progressive neural net outperformed all competing baselines significantly. With an average F1 score of $\sim 99.3\%$, it was able to perform near-perfect task recognition. Further, the progressive net integrated with our task detector plays each MinAtar game comparably well with the same progressive net using task labels. Notably, the baseline algorithms performed poorly in each game except for the final one. The task replay agent received low average rewards on these games, and the EWC agent failed to converge during the second training session and was unable to recover. Additionally, both EWC and task replay required storing extra state data for each task. For EWC, this was a modest 256 records per task, but the replay agent required a more intensive 100,000 records per task to sample from during later training. The quantitative results for this test are shown in Table I (a).

B. Image Repair

Setup. While the MinAtar experiment is a necessary test of the task detector, the distinction between input from tasks is clear. In this experiment, face images are "damaged" in one of several ways and must be repaired. The boundaries between different types of damage are notably fuzzy, and are harder to distinguish compared to game frames. The task set is extended from the exemplar code packaged with the Doric library. The task detector was trained to recognize which set of image processing tasks needed to be applied on a dataset of human faces to uncover a target image. Combined with a progressive system learning the image transformations, this system allows for autonomous image repair.

Two continual learning methods were used alongside the task detector: a progressive neural network and a MoE. The progressive net transfer learns through lateral connections while MoE does so by using the last expert's parameters upon initialization. Additionally, a network was trained with elastic weight consolidation. To act as a baseline continual learner. A baseline classifier was also constructed and trained to evaluate the effects of catastrophic forgetting that would be present without the use of our method alongside the progressive network. It was given the same architecture as the task detector FAEs except with the decoder replaced with a classification head. No measures to reduce catastrophic forgetting were implemented. The classifier also lacked the ability to add new tasks and could not grow progressively, so it was initialized with outputs for each task.

For each image repairer tested, the core network architecture was a convolutional VAE. A subset of the celebA dataset [40] was used for training and testing. Six repair tasks were learned sequentially: denoising; colorizing; inpainting; perspective shift; vertical flip to horizontal flip; invert & flip; and edge image to double flip. A pre-processing step was included for transforming each image to fit the given task — simulating the types of image damage. The testing set consisted of 900 images. Mean squared error was used as a training criterion and for familiarity detection. Image repair task models in this experiment all took the form of VAEs. Dropout and batch normalization were important factors in properly training the networks. Dropout particularly with a drop rate of 0.2 greatly reduced the symptoms of over-fitting. Only a part of the CelebA dataset was used in this work. Images 1 to 30,000 were used to train the networks. Image 30,001 to 30,900 were used for testing and 30,901 to 31,000 were saved for validation.

Results. The task detector show a high level of accuracy operating alongside a progressive neural network and MoE ($\sim$99.6% Accuracy). Combined with downstream tasks, the results were decisive. The EWC repair system performed the worst, and generated strange combinations of facial elements scrambled by catastrophic forgetting. MoE performed much better and in some cases scored the lowest error. However, by inspecting the result images, it is clear that the outputs of this algorithm were fairly low-definition whether paired with task labels or our task detector. The most visually successful repair results came from the progressive neural net and its more-sophisticated transfer learning method. Error was also

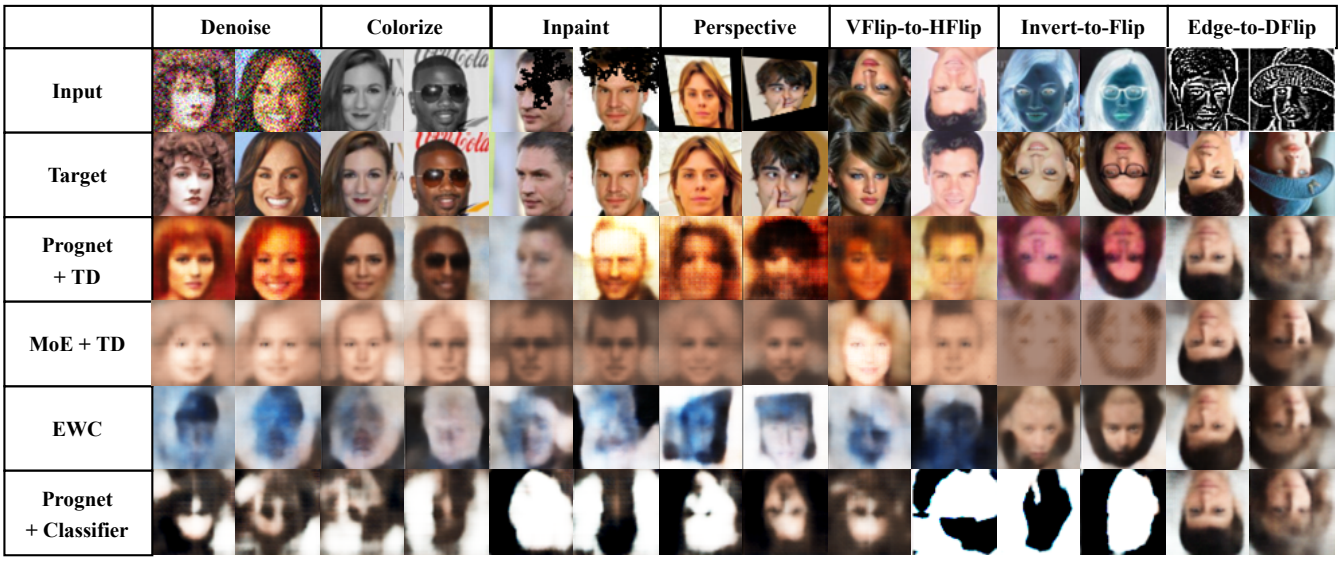

Fig. 3: Selected outputs from the image repair experiment. The first row shows the "damaged" input for each of the 7 tasks. The second row shows the target image that the system should transform the input into. Subsequent rows display the output of each algorithm tested. Training was done in the left-to-right order of the columns (Denoise first, Edge-to-DFlip last). Our "Prognet + TD" creates outputs reasonably close to target for all tasks, indicating that it does not suffer from forgetting even after learning 7 tasks. EWC is able to remember the previous 2 tasks well, while the "Prognet + Classifier" fails on all previous tasks because the classifier succumbs to forgetting.

very consistent between task detector and task label equipped algorithms. Sample image repairs are examined in Fig. 3. The quantitative results for this test are shown in Table I (b).

C. Atari Games

Setup. In [1], the authors train a progressive net to play a set of Atari games to showcase the algorithm's ability to sequentially learn complex reinforcement learning tasks. As a supporting experiment, we ran the task detector alongside a random agent to recognize Atari games from simulated screen frames [41]. Nine games were selected: Breakout, Pong, Space Invaders, Ms Pacman, Assault, Asteroids, Boxing, Phoenix, and Alien. These games can be further classified by type of game (paddle games, ship games, maze games, and fighting games) or by the color of screen assets (i.e. mostly black, mostly colorful). The selected distribution of games serves to test the task detector against similar looking games and similarly colored games, ensuring that the autoencoders utilize both color and content. A baseline classifier like that used in the image repair experiment was used to measure the effects of catastrophic forgetting.

Results. The results of this experiment show that our FAE-based task detector can recognize Atari games at a near-optimal level ($\sim 99.96\%$ F1 score). Conversely, the baseline classifier was severely impacted by catastrophic forgetting and lost the ability to recognize previously learned tasks, only earning an average F1 score of $\sim 70.23\%$. While it is difficult to reconstruct an Atari frame perfectly, it is only necessary that the correct FAE has a closer reconstruction than all of the others — networks that were trained on entirely different data. The quantitative results for this are in Table I (c).

D. Noisy Atari Games

Setup. One desirable property of FAEs as task detectors is their inherent resilience to unexpected noise during operation. This trait is particularly useful in robotic or cyber-physical environments where unexpected noise is a fact of life. To simulate and evaluate these conditions, the task detector was tested again on the Atari environments of the previous experiment. However, this time Gaussian noise was injected into the image during test-time. Three FAE models were tested on the noisy game frames: the standard autoencoder; a variational autoencoder; and an adversarial autoencoder. All models were trained on non-noisy data, and tested on 100 frames of noisy data to determine the noise resistance of these models in relation to the task detector's classification efficacy.

Results. While all task detector architectures outperform the forgetting-sensitive baseline significantly, the default FAE does appear to suffer when encountering unexpected noise. Fortunately, this potential limitation is nullified by the VAE-based FAE, which performs another near-perfect classification even with severe noise present in the input. The noise resilience shown here is a good sign of the task detector's ability to perform in real-world scenarios. The quantitative results for this test are shown in Table I (d).

V. Conclusion

In this paper, we introduced a task detection system using familiarity autoencoders to predict task information from input patterns while maintaining immunity to catastrophic forgetting. The results of our experiments with MinAtar, Atari, and image repair demonstrate that our task detector is capable of recognizing tasks with great accuracy and combines well with downstream task models like progressive neural networks. The concept behind the task detector could be applied more widely, as it is essentially a special type of classifier with built-in forgetting immunity.

For the future of progressive systems in continual learning to be a rich one, adding greater autonomy during training and operation will be paramount. One future avenue of exploration is the potential of learning not only the task labels, but also the task boundaries during training. This could potentially be done with a heuristic that predicts the behavior of an FAE when it detects a new task, without skipping the early stages of a current task. Even a training step counter and threshold could be used to attempt this, though more-intelligent solutions yet to be developed would likely be more effective.

VI. Acknowledgements

Thank you to Md Masudur Rahman and all the anonymous reviewers. This research was supported by NSF grants IIS-1850243, CCF-1918327.

References

[1] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *CoRR*, vol. abs/1606.04671, 2016.

[2] M. D. Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3366–3385, 2022.

[3] T. L. Hayes, G. P. Krishnan, M. Bazhenov, H. T. Siegelmann, T. J. Sejnowski, and C. Kanan, "Replay in deep learning: Current approaches and missing biological elements," *Neural Computation*, vol. 33, no. 11, pp. 2908–2950, 2021.

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

[5] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[6] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," in *ICLR (Poster)*. OpenReview.net, 2018.

[7] R. K. Pathak and V. Yadav, "Improvised progressive neural network (ipnn) for handling catastrophic forgetting," in *ICESC*, 2020, pp. 143–148.

[8] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, "Pathnet: Evolution channels gradient descent in super neural networks," *CoRR*, vol. abs/1701.08734, 2017.

[9] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

[10] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *ICML*, vol. 70. PMLR, 2017, pp. 3987–3995.

[11] E. Ikonomovska, J. Gama, and S. Džeroski, "Learning model trees from evolving data streams," *Data mining and knowledge discovery*, vol. 23, no. 1, pp. 128–168, 2011.

[12] J. Gao, W. Fan, J. Han, and P. S. Yu, "A general framework for mining concept-drifting data streams with skewed distributions," in *SDM*. SIAM, 2007, pp. 3–14.

[13] A. Wibisono, W. Jatmiko, H. A. Wisesa, B. Hardjono, and P. Mursanto, "Traffic big data prediction and visualization using fast incremental model trees-drift detection (fimt-dd)," *Knowledge-Based Systems*, vol. 93, pp. 33–46, 2016.

[14] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, 2021.

[15] R. Hadsell, D. Rao, A. A. Rusu, and R. Pascanu, "Embracing change: Continual learning in deep neural networks," *Trends in Cognitive Sciences*, vol. 24, no. 12, pp. 1028–1040, 2020.

[16] M. D. Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3366–3385, 2022.

[17] G. M. van de Ven and A. S. Tolias, "Three scenarios for continual learning," *CoRR*, vol. abs/1904.07734, 2019.

[18] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *NIPS*, 2017, pp. 6467–6476.

[19] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with A-GEM," in *ICLR (Poster)*. OpenReview.net, 2019.

[20] 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.

[21] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *ECCV*, vol. 11207, 2018, pp. 144–161.

[22] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, 2018.

[23] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," *arXiv preprint arXiv:1705.08690*, 2017.

[24] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne, "Experience replay for continual learning," in *NeurIPS*, 2019, pp. 348–358.

[25] J. Xu and Z. Zhu, "Reinforced continual learning," *arXiv preprint arXiv:1805.12369*, 2018.

[26] T. J. Draelos, N. E. Miner, C. C. Lamb, J. A. Cox, C. M. Vineyard, K. D. Carlson, W. M. Severa, C. D. James, and J. B. Aimone, "Neurogenesis deep learning: Extending deep networks to accommodate new classes," in *IJCNN*. IEEE, 2017, pp. 526–533.

[27] A. A. Rusu, M. Vecerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," in *CoRL*, vol. 78, 2017, pp. 262–270.

[28] J. Gideon, S. Khorram, Z. Aldeneh, D. Dimitriadis, and E. M. Provost, "Progressive neural networks for transfer learning in emotion recognition," in *INTERSPEECH*. ISCA, 2017, pp. 1098–1102.

[29] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[30] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[31] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, 2014, pp. 4–11.

[32] D. C. Mocanu and E. Mocanu, "One-shot learning using mixture of variational autoencoders: a generalization learning approach," in *AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018, pp. 2016–2018.

[33] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert gate: Lifelong learning with a network of experts," in *CVPR*. IEEE Computer Society, 2017, pp. 7120–7129.

[34] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014.

[35] A. Makhzani, J. Shlens, N. Jaitly, and I. J. Goodfellow, "Adversarial autoencoders," *CoRR*, vol. abs/1511.05644, 2015.

[36] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial networks," *CoRR*, vol. abs/1406.2661, 2014.

[37] K. Young and T. Tian, "Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments," *arXiv preprint arXiv:1903.03176*, 2019.

[38] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *NeurIPS*, 2019, pp. 8024–8035.

[39] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[40] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *ICCV*. IEEE Computer Society, 2015, pp. 3730–3738.

[41] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *CoRR*, vol. abs/1606.01540, 2016.