Task Detectors for Progressive Systems

By Maxwell J. Jacobson

Abstract

- While methods like learning-without-forgetting[1] and elastic weight consolidation[2] accomplish high-quality transfer learning while mitigating catastrophic forgetting[3,4], progressive techniques such as Deepmind's progressive neural network[5] accomplish this while completely nullifying forgetting.
- However, progressive systems like this strictly require task labels during test time.
- In my paper, *Task Detectors for Progressive Systems*, I introduce a novel task recognizer built from anomaly detection autoencoders that is capable of detecting the nature of the required task from input data.
- Alongside a progressive neural network or other progressive learning system, this task-aware network is capable of operating without task labels during run time while maintaining any catastrophic forgetting reduction measures implemented by the task model.

Contributions

1. Introduce a novel extension to progressive neural networks that allows for test time task recognition without introducing catastrophic forgetting.
2. Demonstrate the quality of the extension on two difficult task sets.
3. Contribute towards an eventual full continual learning system.

Background

Types of Learning

Three main areas of machine learning:

1. Supervised
2. Unsupervised
3. Reinforcement

Self-supervised learning is a set of methods that allows an unsupervised system to learn in a supervised way by exploiting features of the data.

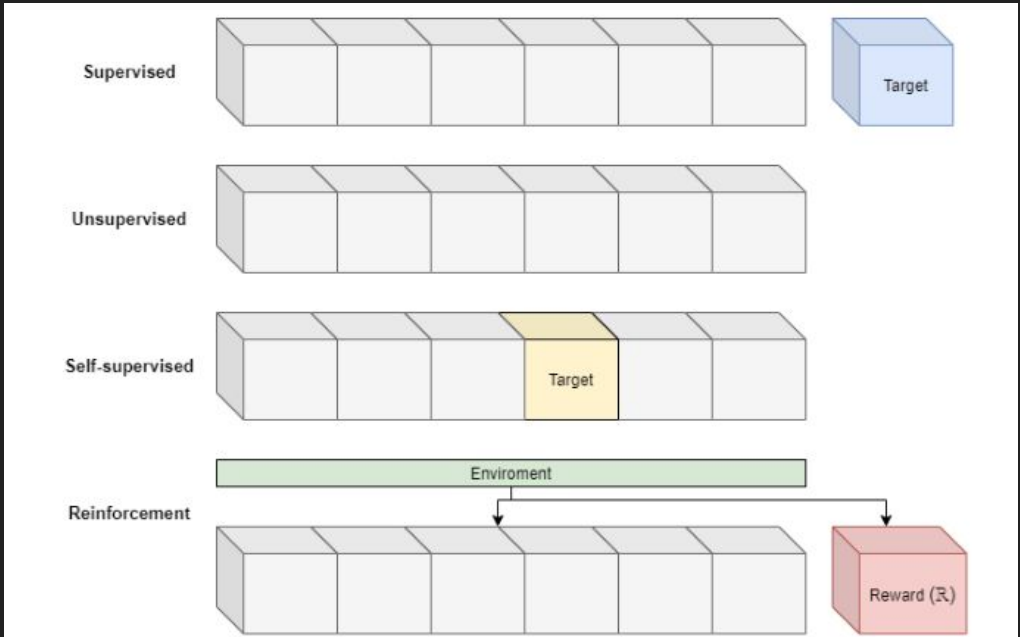


Figure 1. A visual representation of data supplied for supervised, unsupervised, self-supervised, and reinforcement learning.

Types of Learning

All of these paradigms posit data to be used in solving a task, with varying degrees of structure.

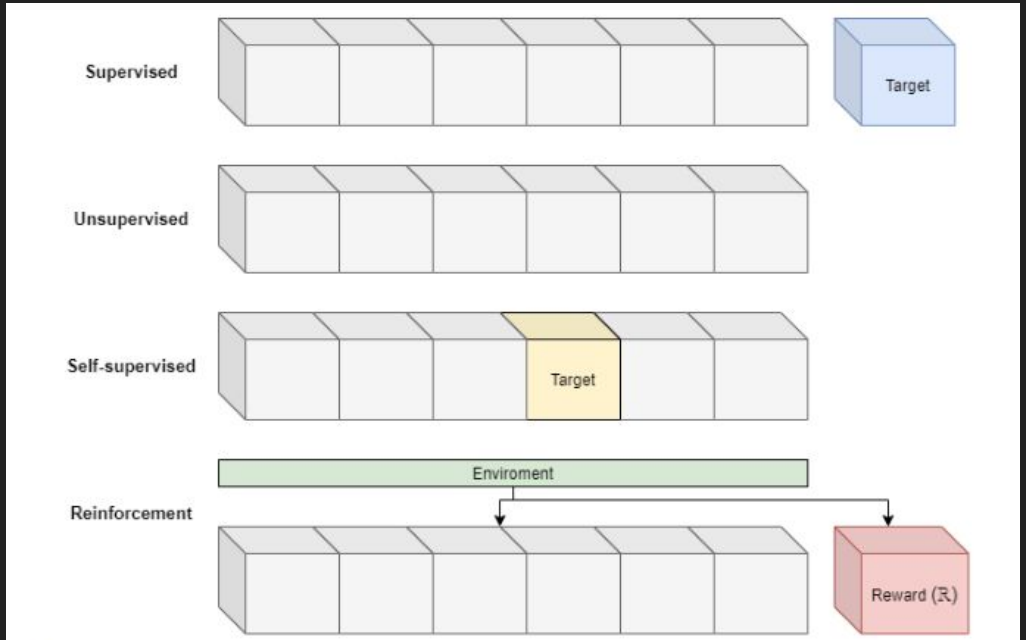


Figure 1. A visual representation of data supplied for supervised, unsupervised, self-supervised, and reinforcement learning.

Transfer Learning

Transfer learning is the process of transferring the competence from one machine learning system learning a task to another system learning a related task. Transfer learning is deeply entangled with the concept of continual learning — that is, ML systems that are capable of constantly improving while remaining stable on previous tasks.

The most common methods for this include representation learning which is only possible when input data generalizes well to all selected tasks, and finetuning — a method where learned parameters from one task are simply modified for the next task.

Catastrophic Forgetting

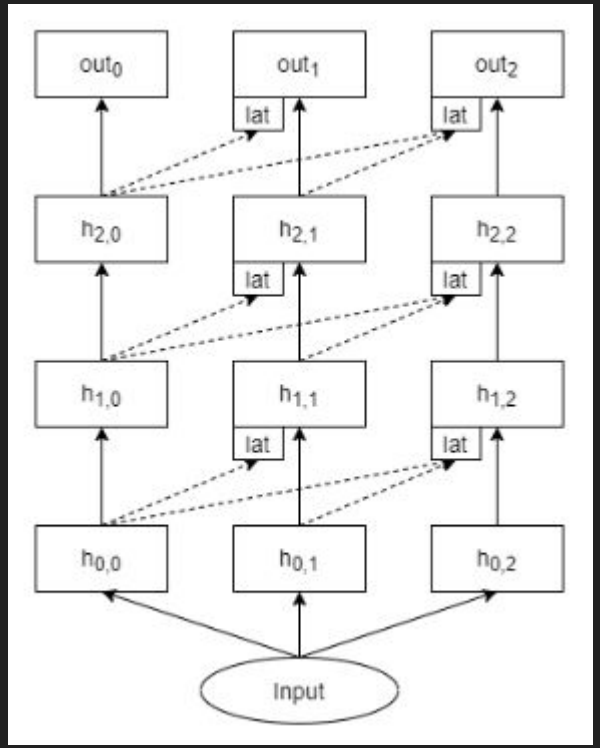
Catastrophic forgetting[3,4]:

- A deleterious event that can occur within a trained neural net during finetuning.
- Occurs when important parameters within the network are changed to fit the new data, compromising the network's ability to handle the old data.
- Ideally, a network would retain its competence on prior tasks by encoding new information within unused parameters or by changing important parameters in a way that generalizes to the new task.
- Using standard gradient-based learning methods, this is often impossible.

Progressive Systems

Progressive neural networks[5] build new columns for new tasks, creating lateral connections to allow transfer learning and freezing trained columns to immunize them to catastrophic forgetting.

Tasks are considered to be discreet with the programmer supplying a task label. New tasks can only be added once the previous task has converged and is done training.

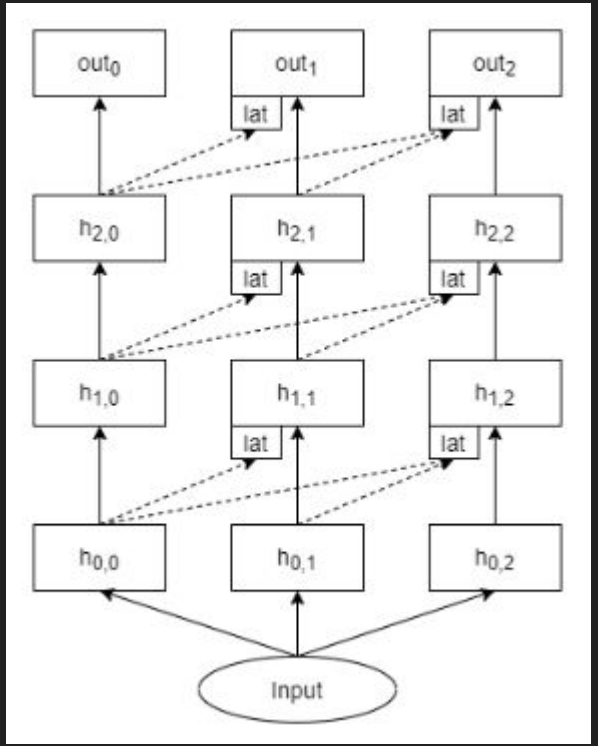


Progressive Systems

$$h_{0,i} = f\left(W_{0,i} \cdot h_{0,i-1} + b_{0,i}\right)$$

$$h_{k,i} = f\left(W_{k,i} \cdot h_{k,i-1} + \sum_{j<k} \left(U_{j,i} \cdot h_{j,i-1}\right) + b_{k,i}\right)$$

The equations governing each block in the progressive net in the initial column and in subsequent columns.



Progressive Systems

Progressive neural networks have proven to be valuable in several types of learning:

- Atari games[5]
- Emotion detection[6]
- Sim-to-robot transfer learning[7]

Anomaly Detection

Anomaly detection — also called novelty detection — is the task of identifying anomalies, outliers, or unusual data records among a dataset.

This is similar to one-class classification, in which ML systems must decide whether a data record is part of a single class or not.

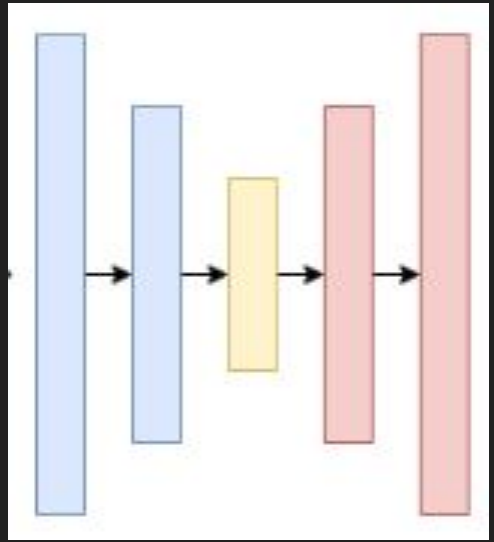
Anomaly Detection

Common anomaly detection techniques:

- One-class SVMs[8]
- One-class neural networks[9]
- Unsupervised methods
- Reconstructive autoencoders

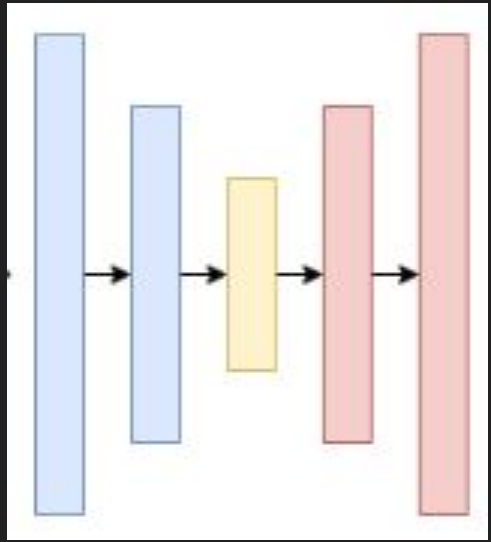
Autoencoders

- Encoder -- Learns to encode/compress input to fit the latent vector.
- Latent vector -- The middle of the network that acts as a “bottleneck” for data, forcing the network to learn abstractions of the dataset.
- Decoder -- reconstructs the input to its original form or the desired transformation.

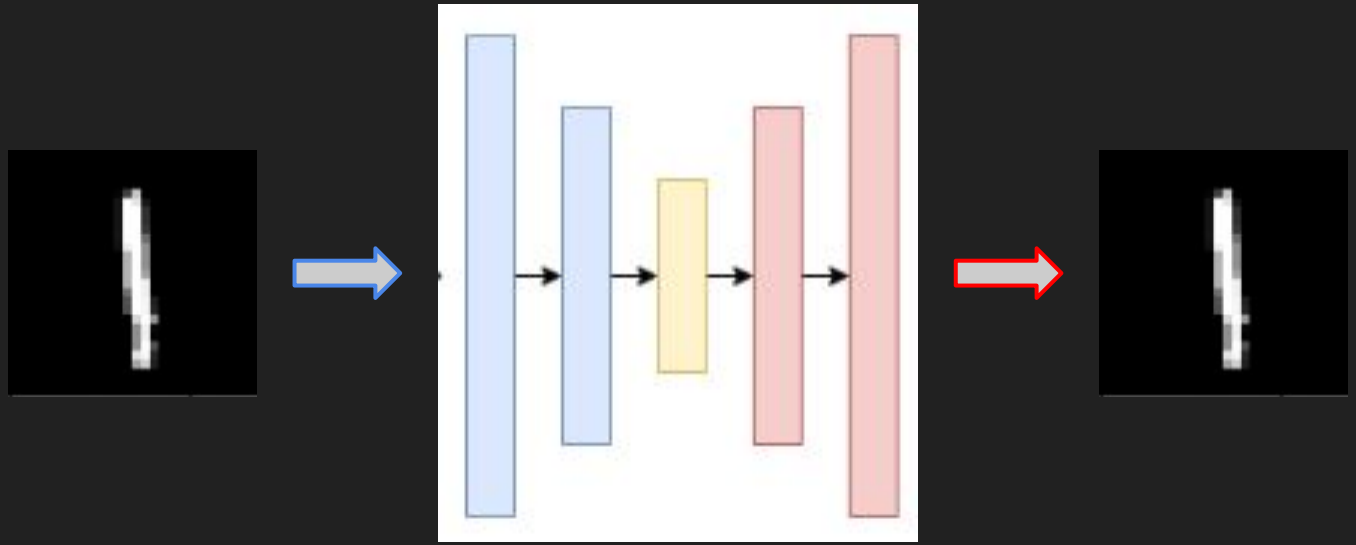


Autoencoders

- Encoder → $E(X) = X_L$
- Latent vector → $length(X_L)$
- Decoder → $D(X_L) = X$ or $D(X_L) = f(X)$



Anomaly Detection Autoencoders



* anomaly detection autoencoders [10], MNIST [11]

Anomaly Detection Autoencoders

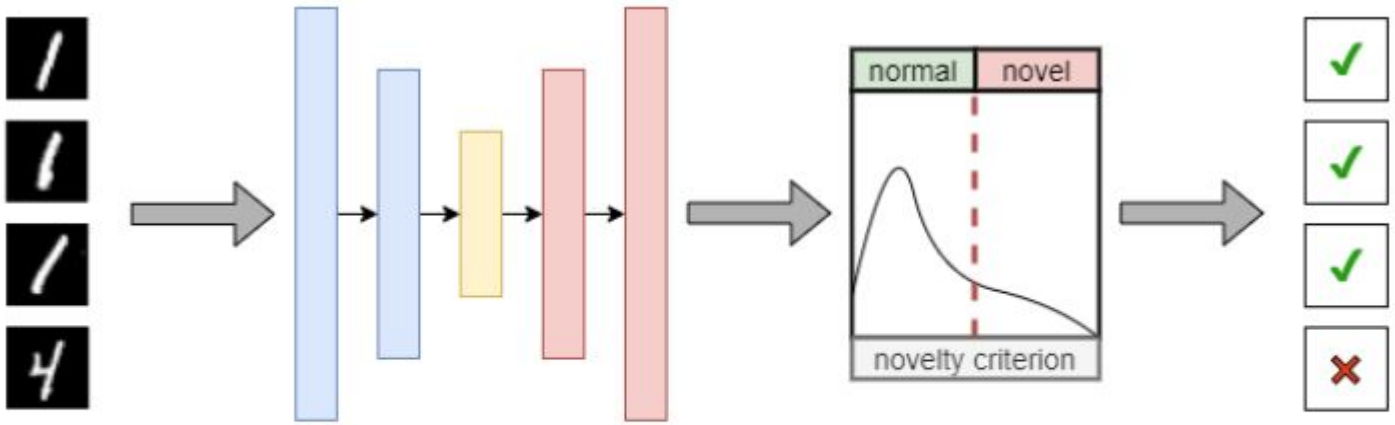


Figure 7. Diagram of an anomaly detection autoencoder learning to reconstruct handwritten '1' digits from the MNIST dataset. The autoencoder reconstructs the data and checks against a threshold of the novelty criterion to classify the data as normal or novel.

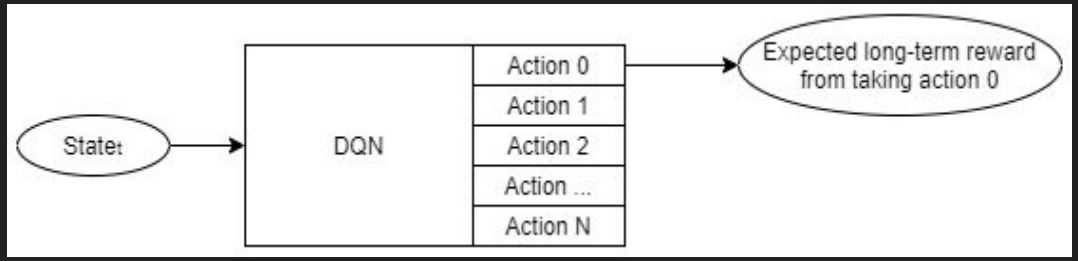
* anomaly detection autoencoders [10], MNIST [11]

Methodology

Example Problem (3 Atari Games)

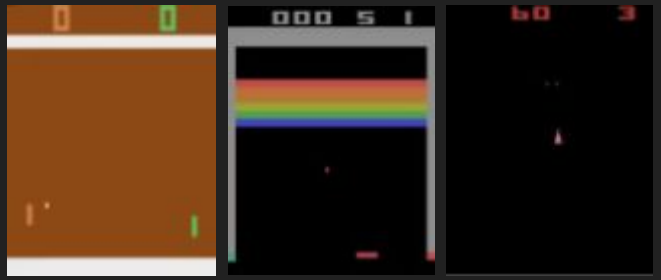
We have three atari games we want to learn how to play. We want to transfer learn each game from the last game, and we must train sequentially.

For the task model, we use a simple reinforcement learning technique. A type of value iteration known as Deep Q-Learning[12] in which the expected reward is predicted given the current state and a potential action according to the action-value Bellman equation.

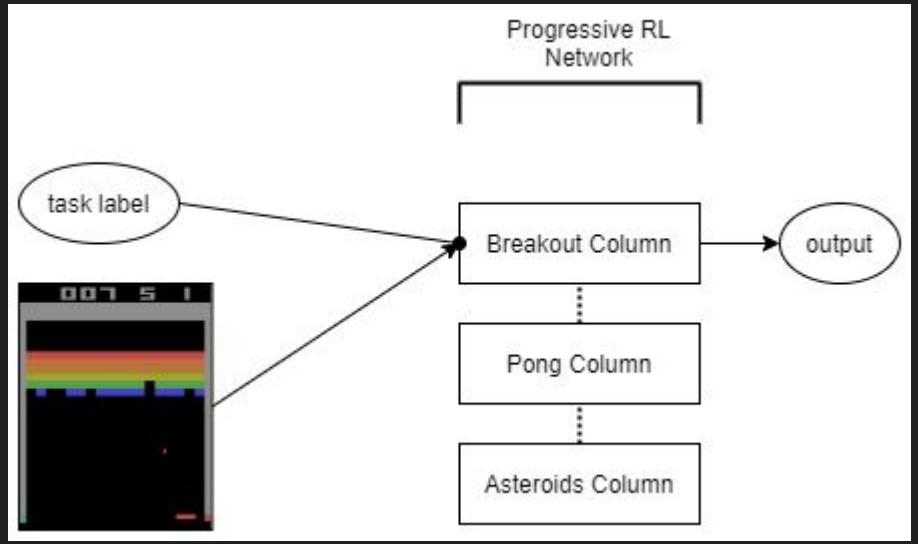


$$Q_{\pi}(s_t, a_t) = E(r_{t+1} + \gamma * r_{t+2} + \gamma^2 * r_{t+3} + ... \mid s_t, a_t)$$

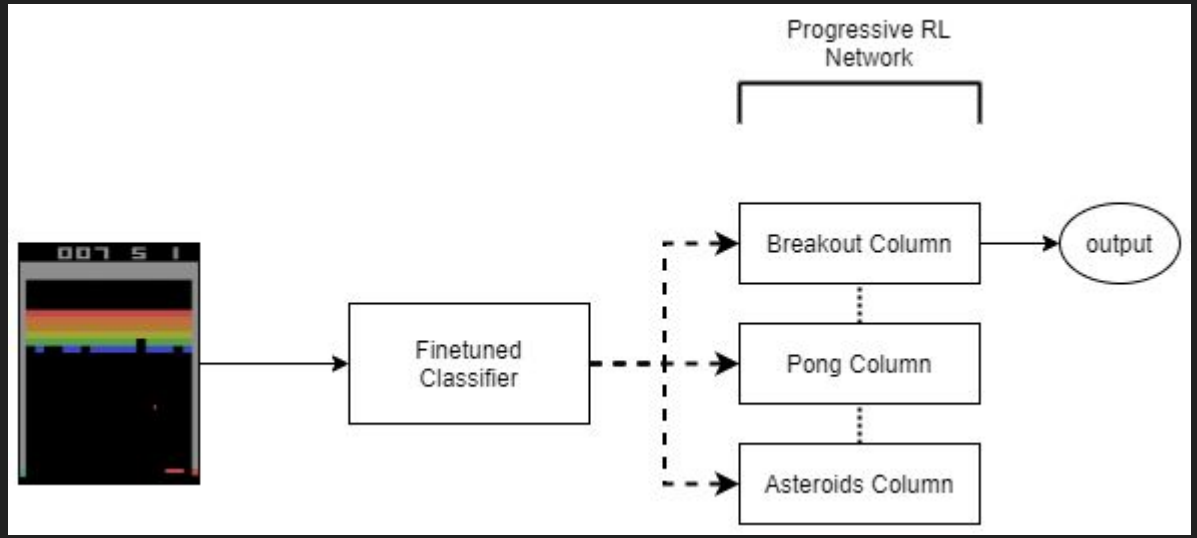
Figure 3. The Q function mapping state and action to reward. γ is a $(0.0, 1.0)$ discount factor that reduces the importance of rewards farther in the future.



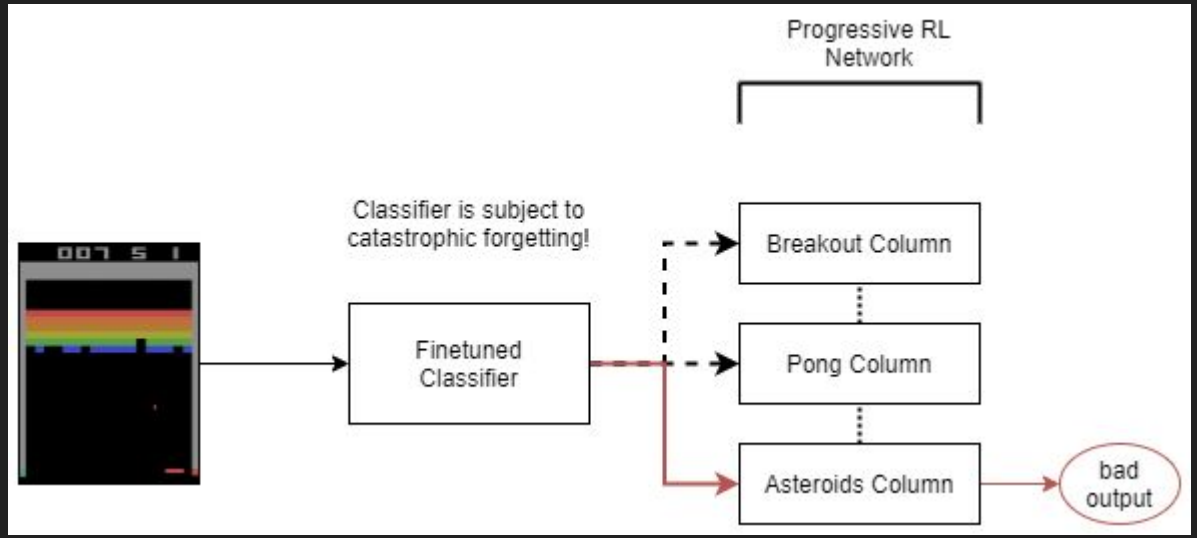
Comparisons (Vanilla Progressive Net)



Comparisons (Classifier)

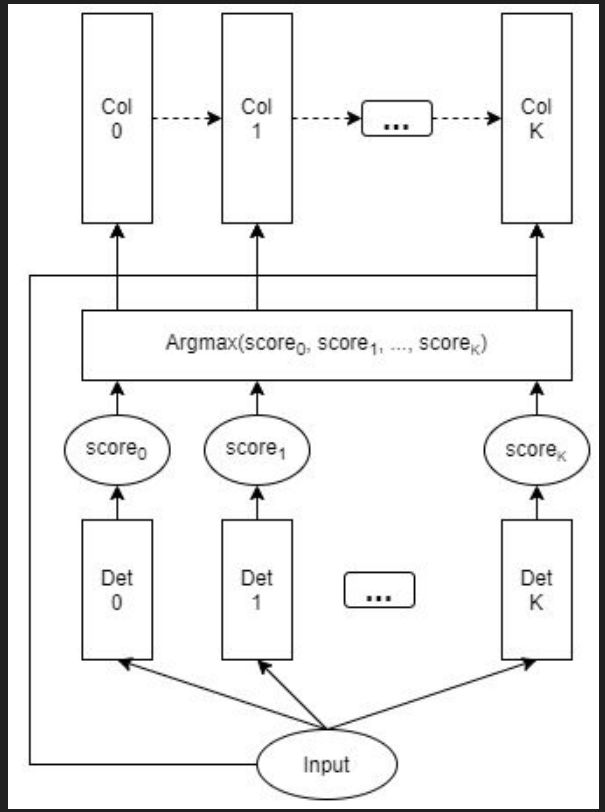


Comparisons (Classifier)



Task Detector + Progressive Network

- Progressive neural network as a set of task models.
- Array of independent anomaly detectors indexed to progressive network columns.
- In my implementation, these anomaly detectors are autoencoders.
- Scores can also be easily normalized for unbalanced tasks, but this was inconsequential on our experiments.



Task Detector Training

- Progressive net column is trained alongside its anomaly detector on the same data.
- Once the task detector and column converge, neither is modified again.

Algorithm: Training Task Detector with Progressive Net

Input: Progressive neural network P, task detector D, and task set T of size K.

Initialize networks P and D

for $k \leftarrow 0$ ***to*** K **do**

- $t \leftarrow T_k$;
- Add column k to P;
- Add autoencoder k to D;
- Train P_k on t;
- Train D_k to reconstruct inputs from t;
- Freeze P_k and D_k;

end

return P and D;

Task Detector Testing

- Input is fed to each autoencoder in the task detector.
- Using the novelty criterion, a score is computed, also indexed to the associated progressive column.
- An argmax is calculated.
- The index generated by the argmax is used to select the column.

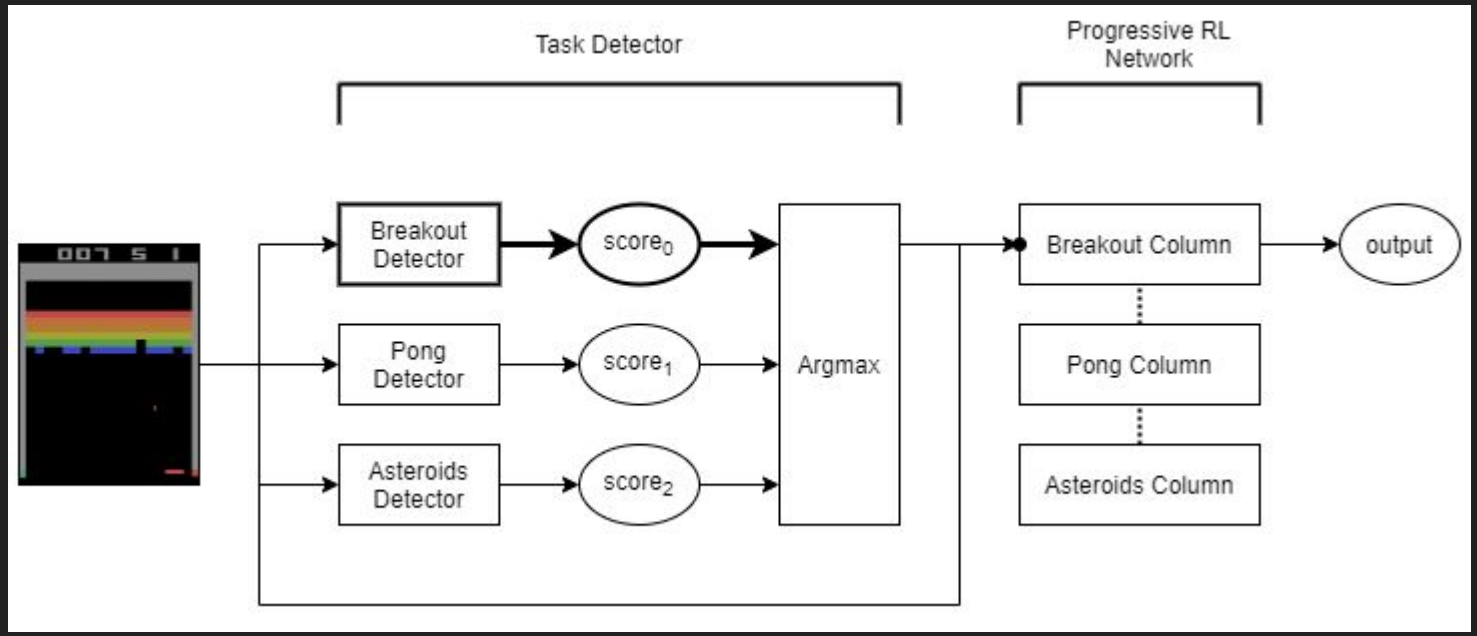
$$\hat{k} = argmax_{\mathrm{i} \in \{0,\ldots,K\}}(score(x, d_{\mathrm{i}}(x)))$$

Algorithm: Testing Task Detector with Progressive Net

Input: Progressive neural network P, task detector D, and input data x.

```
maxK ← null;
maxScore ← −infinity;
for k ← 0 to K do
    score ← −novelty_criterion(D_k(x));
    if score > maxScore then
        maxK ← k;
        maxScore ← score;
    end
end
return P_maxK(x);
```

Comparisons (Task Detector)

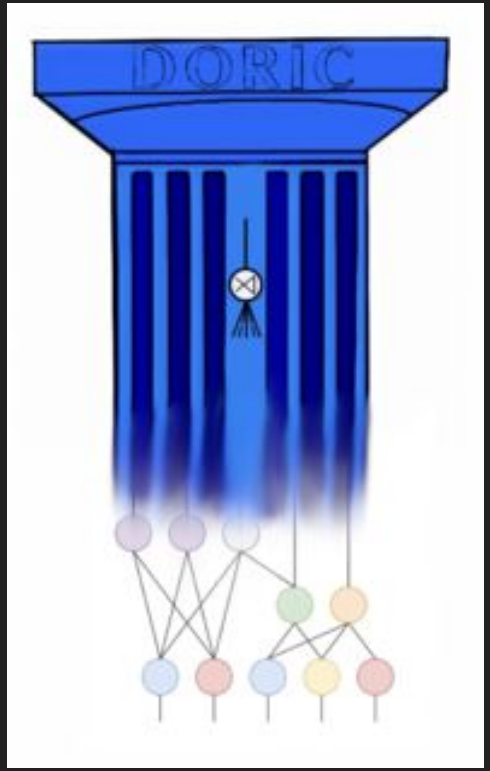


Experiments

Doric

All progressive neural nets used in this project were implemented using PyTorch and the Doric library.

Our Towards Data Science article on making Doric can be found here: https://towardsdatascience.com/progressive-neural-networks-explained-implemented-6f07366d714d



Atari Games

Atari games from OpenAI gym[13] were learned one at a time and then tested on 501 frames from each game.

Input to the network is a single frame of the screen. The next frame is also given to calculate KAMSE.

Atari Games

Games included:

- Breakout
- Pong
- Space Invaders
- Ms Pacman
- Assault
- Asteroids
- Boxing
- Phoenix
- Alien

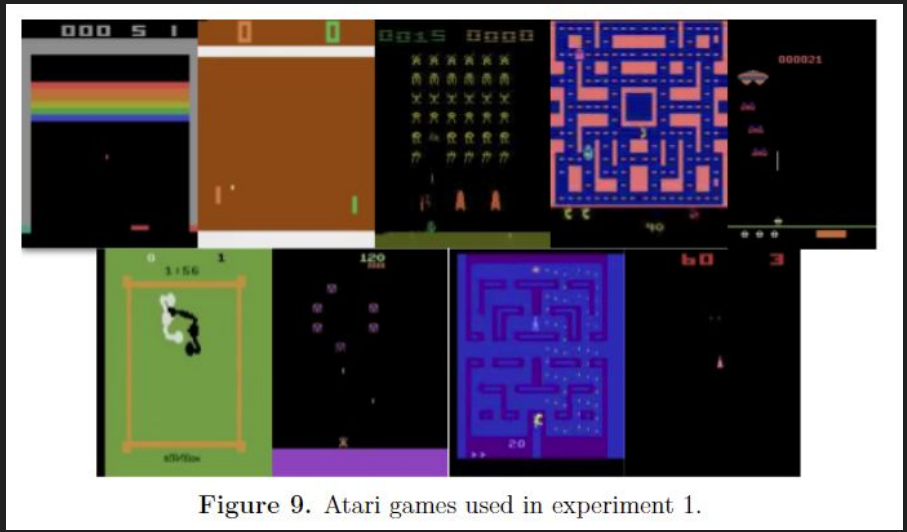


Figure 9. Atari games used in experiment 1.

Kinetically-adjusted Mean Squared Error (KAMSE)

A novel autoencoder loss function that puts more attention on pixels that have changed between two frames of a game.

$$KAMSE(y_0, \hat{y_0}, y_1) = MSE(y_0, \hat{y_0}) + \kappa * MSE(M_{y_0,y_1} * (y_0), M_{y_0,y_1} * (\hat{y_0}))$$

$$M_{y_0,y_1,\mathrm{i}} = \begin{cases} 0 & y_{0,\mathrm{i}} - y_{1,\mathrm{i}} = 0, \\ 1 & otherwise. \end{cases}$$

CelebA Image Tasks

Several image tasks were learned on the celebA face image dataset[14].

5000 images were tested for each transformation.

Tasks included:

- simple reconstruction
- noise removal
- Colorizing
- inpainting

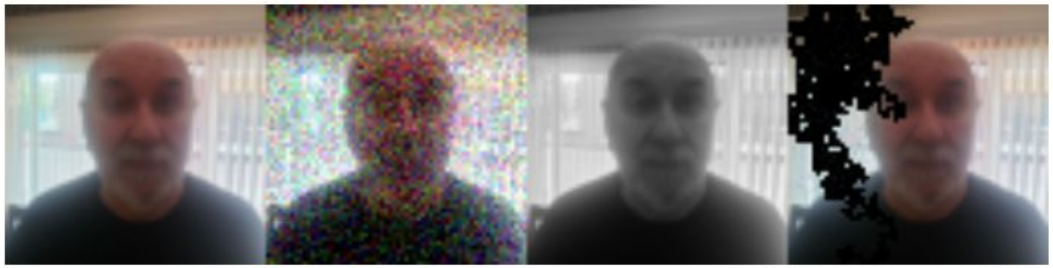


Figure 11. Demonstration of image tasks. From left to right: reconstruction, denoising, colorizing, and inpainting.

Detector Architecture

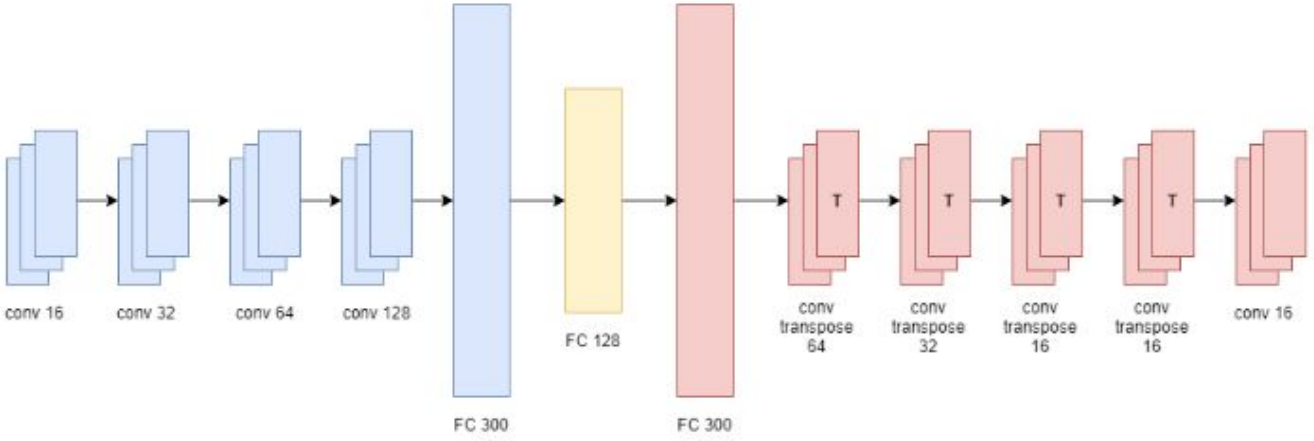


Figure 10. Architecture of an atari game anomaly detector. All convolutional and transposed convolutional layers have a kernel size of 3, a stride of 1, and a padding of 1. Leaky ReLU acts as the activation function for all hidden layers. Tanh acts as the activation for the output layer.

Detector Architecture

This same architecture is used for the CelebA image transformation experiment, except with a reduced latent vector of size 64.

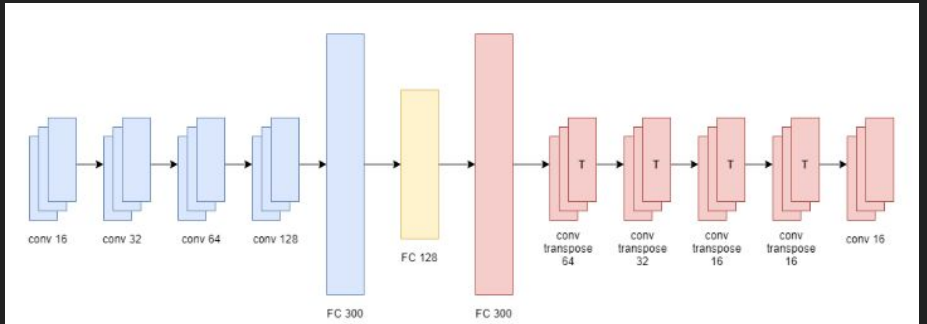
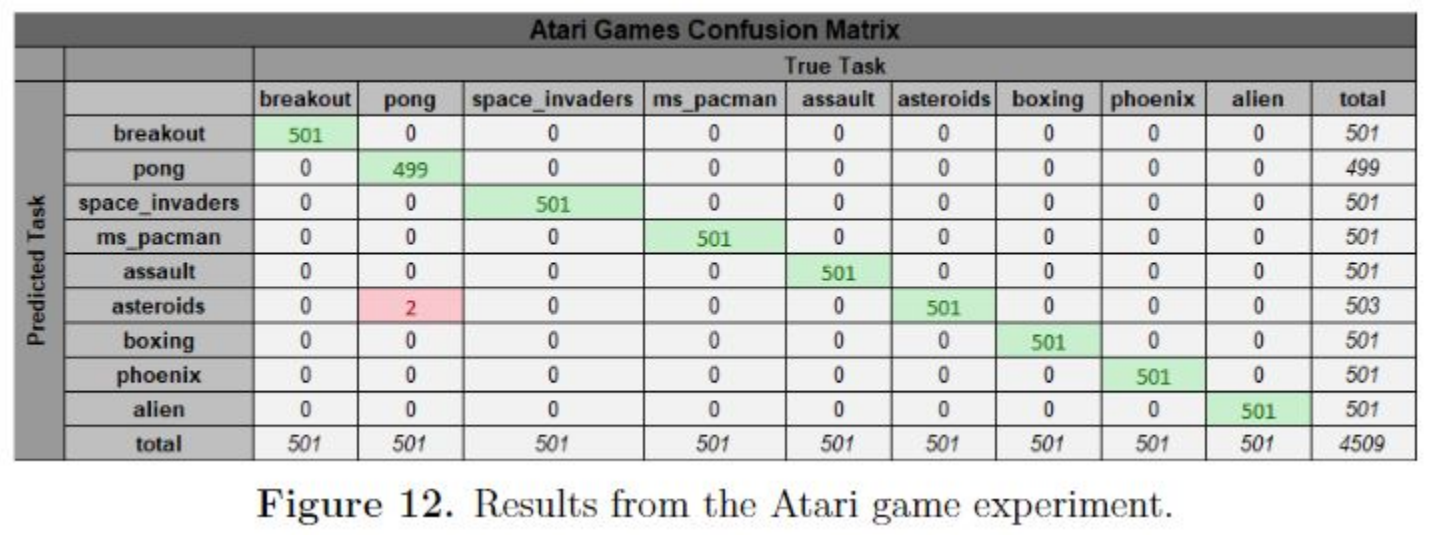


Figure 10. Architecture of an atari game anomaly detector. All convolutional and transposed convolutional layers have a kernel size of 3, a stride of 1, and a padding of 1. Leaky ReLU acts as the activation function for all hidden layers. Tanh acts as the activation for the output layer.

Results

Atari Games



Atari Games Confusion Matrix

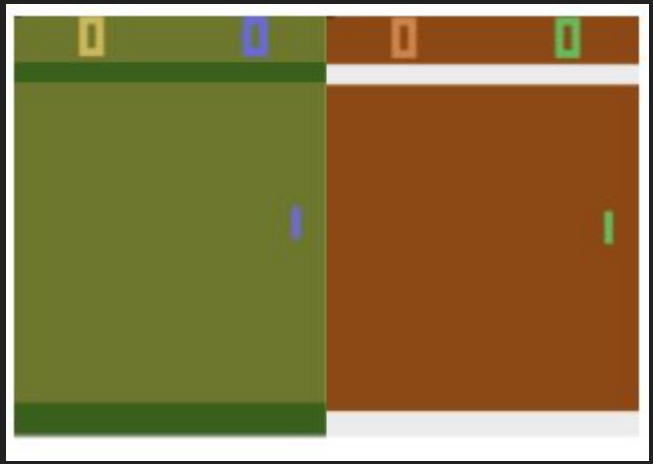
Predicted Task \ True Task	breakout	pong	space_invaders	ms_pacman	assault	asteroids	boxing	phoenix	alien	total
breakout	501	0	0	0	0	0	0	0	0	*501*
pong	0	499	0	0	0	0	0	0	0	*499*
space_invaders	0	0	501	0	0	0	0	0	0	*501*
ms_pacman	0	0	0	501	0	0	0	0	0	*501*
assault	0	0	0	0	501	0	0	0	0	*501*
asteroids	0	2	0	0	0	501	0	0	0	*503*
boxing	0	0	0	0	0	0	501	0	0	*501*
phoenix	0	0	0	0	0	0	0	501	0	*501*
alien	0	0	0	0	0	0	0	0	501	*501*
total	*501*	*501*	*501*	*501*	*501*	*501*	*501*	*501*	*501*	*4509*

Figure 12. Results from the Atari game experiment.

Average F1 Score: 0.9996

Graphical Quirk

The two misclassified examples are likely artifacts of the Pong first-frame graphical quirk.



CelebA Image Tasks

Face Image Tasks Confusion Matrix with Batch_Size = 8						
		True Task				
Predicted Task		reconstruction	denoising	colorizing	inpainting	total
	reconstruction	4984	0	0	0	*4984*
	denoising	0	5000	0	0	*5000*
	colorizing	0	0	5000	0	*5000*
	inpainting	16	0	0	5000	*5016*
	total	*5000*	*5000*	*5000*	*5000*	*20000*

Figure 14. Results from the face image tasks experiment.

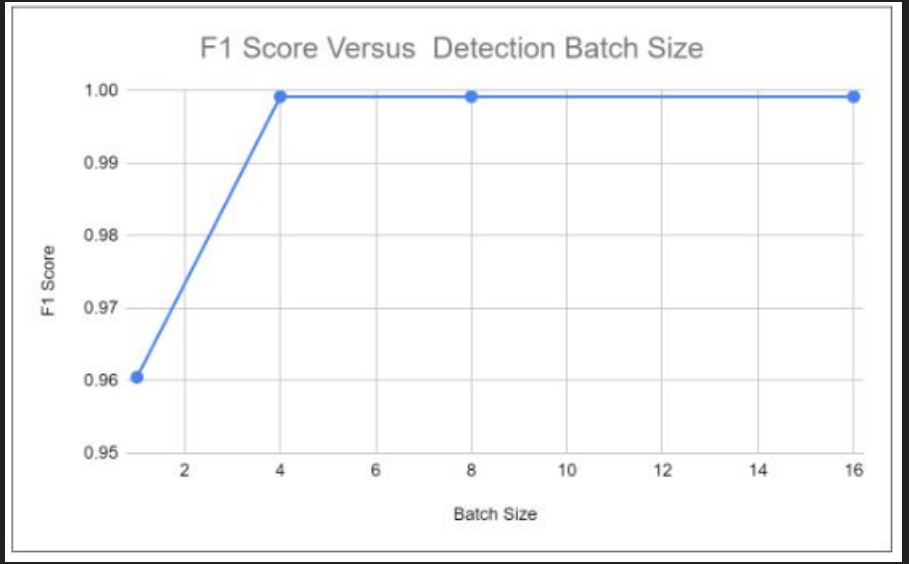
Average F1 Score: 0.9992

Batch Size

Greater batch sizes correlated with higher F1 score.

More examples allow for small mistakes by the detectors or criterion, adding redundancy and increasing the difference between novelty scores.

Batches purely of the same task are not always possible.



Discussion

Task Detector Limitations

- Relies on the assumption that the task is recognizable from only its input data.
  - E.G. -- Pong with inverted controls.
  - Can be solved with tweeks to data representation or architecture.
- Memory intensive. 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.
- Extra training time required to train the detectors.
  - Usually not a big difference.

Task Detector Strengths

- As it uses self-supervised learning, it will rarely require as much data to train as its task network — a comparison that is all the more stark when applied to reinforcement learning tasks
- It is also likely to be finished training well before the task network on any sufficiently difficult task, as reconstruction of inputs is usually a relatively simple job.
- Because the task model and task detector do not interact during training, they can easily be trained in parallel.
- The Task Detector is error-resilient as the correct anomaly detector only needs to perform better than the incorrect detectors, which are not trained on the same data.

Future Works: Progressive Task Detectors

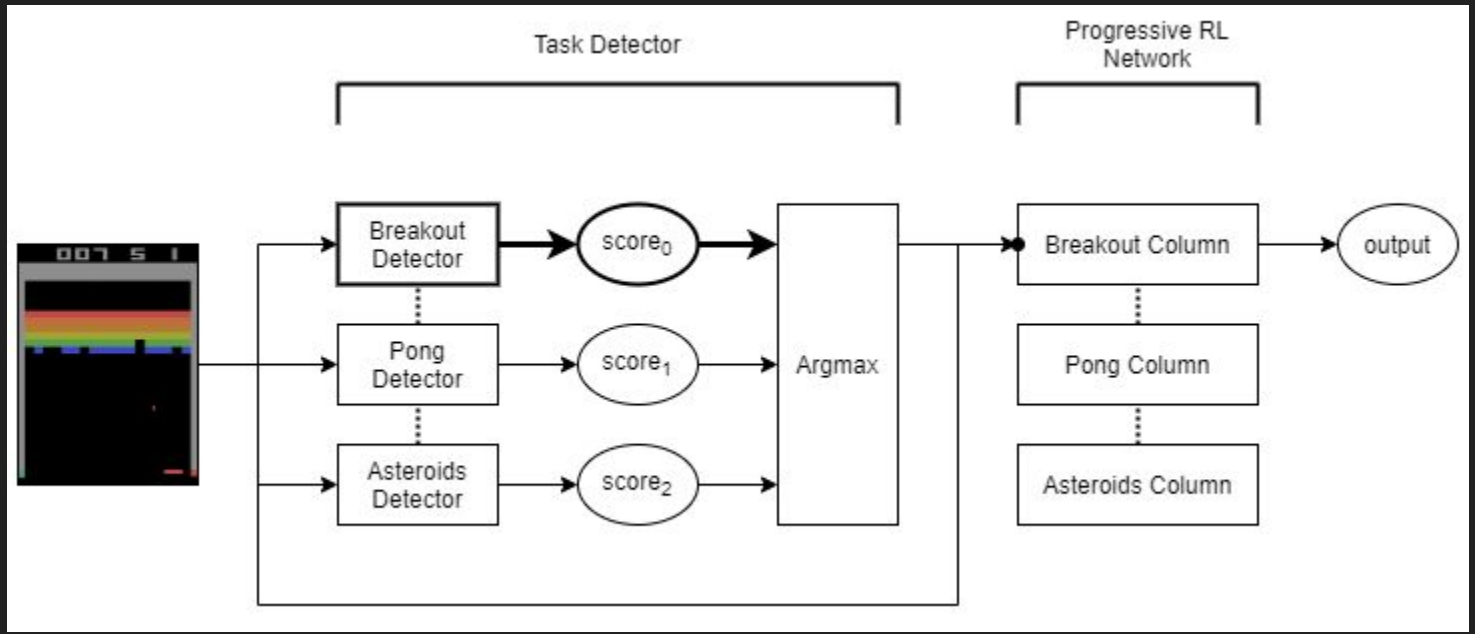
Why not link the anomaly detectors with a progressive network?

This would allow for transfer learning within the task detector.

By using many of the same parameters, different anomaly detectors would likely retain some reconstruction ability from previously learned tasks.

It would also introduce even more restrictive memory requirements — one of the main reasons the variant wasn’t tested.

Future Works: Progressive Task Detectors



Conclusions

“Continual learning, the ability to accumulate and transfer knowledge to new domains, is a core characteristic of intelligent beings. Progressive neural networks are a stepping stone towards continual learning” - *Progressive Neural Networks*, 2016

By expanding on the test time autonomy of progressive networks, allowing the system to identify the necessary column to run while maintaining its immunity to catastrophic forgetting, the autoencoder task detector succeeds in being another stepping stone towards this worthy goal.

Code Repository

The code used in this study can be found at:

www.github.com/arcosin/Task-Detector

The Doric library can be found at:

www.github.com/arcosin/Doric

Acknowledgments

Thank you to Dr. Gustavo Rodriguez-Rivera for guidance and academic support, and to Case Wright for assistance with autoencoders and KAMSE loss.

References

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

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

[3] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In Psychology of learning and motivation, volume 24, pages 109–165. Elsevier, 1989.

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

[5] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks, 2016.

[6] John Gideon, Soheil Khorram, Zakaria Aldeneh, Dimitrios Dimitriadis, and Emily Mower Provost. Progressive neural networks for transfer learning in emotion recognition, 2017.

[7] Andrei A. Rusu, Mel Vecerik, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets, 2016.

References

[8] Bernhard Schölkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. InNIPS'99: Proceedings of the 12th International Conference on Neural Information Processing Systems, volume 12,pages 582–588, 01 1999.

[9] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Anomaly detection using one-class neural networks, 2018.

[10] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with non-linear dimensionality reduction. In Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis, pages 4–11, 2014.

[11] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

[13] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. cite arxiv:1606.01540.

[14] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In Proceedings of International Conference on Computer Vision (ICCV),December 2015.