Continual Learning: 3rd CLVISION CVPR Workshop Task 1 Instance Classification

Noah Corbett
Department of Computer Science, University of Saskatchewan
ndc864@usask.ca

Abstract

Continual learning is a branch of deep learning that aims to mitigate catastrophic forgetting, but still allow the model to change and adapt to new data it is being trained on via learning strategies that either remember past data or parameter regularization schemes. The first challenge track of the CVPR CLVISION workshop is focused on class incremental learning which is learning new classes in separate phases also known as experiences. Only samples of the current experience are available to be trained on. For this challenge track, there is a template provided with several commonly used strategies for continual learning. Thus, this project aims to see if I can add another strategy to a previous submission to make the model perform better. The GitHub link is provided here: https://git.cs.usask.ca/ndc864/cmpt-489-continual-learning

1. Introduction

The first challenge track for CLVISION is an instance classification track. This track focuses on how to mitigate catastrophic forgetting while going through a series of learning different classifications. The challenge tracks also come with a devkit that is provided that also contains commonly used continual learning strategies that try to combat catastrophic forgetting that is readily available as plugins [1]. These plugins are from a library called Avalanche which is an End-to-End Continual Learning Library Based on PyTorch [1]. Finding strategies to mitigate catastrophic forgetting will allow models to be able to run and learn in more dynamic and changing environments. Which allows more applications of deep learning models in the real world. This is challenging because the current models for deep learning usually do training and testing in two different static phases which is hard to transfer to a dynamic environment. Thus, improving the effectiveness of strategies or finding the right combination of them for the task will be beneficial for many areas, both research and real-world applications.

1.1. Instance Classification Challenge Track

The continual learning instance classification track divides the classes into 15 different training experiences. After each training experience the model will be tested on the full test set to compute the TOP-1 accuracy at each experience which is then averaged. The classes are ordered using a seed that came with the dev environment which you are not allowed to change. The other important and notable rules of the challenge track are as follows

- The benchmark creation procedure can't be changed. Instances will be loaded as 224x224 RGB images. A different input size can be used in the solution, but the image must first be loaded as 224x224.
- The size of each model obtained after each training iteration (including the final model) is limited to 70M parameters (defined as the number of parameters, not its size in MBytes).
- Model initialization can be done by randomly initializing weights or by pretraining using the ImageNet-1K (ImageNet 2012) dataset (recommended). Apart from the pre-trained weights, the solution must not access/use data from the pretraining dataset.
- Solutions can exploit a replay buffer containing data coming from up to 3500 training samples (image + label). The buffer must be initially empty and can be populated using data from the current experience only.
- The maximum allowed execution time for the whole solution (training + test) is 12 hours.

2. Methods

The methods section will cover the 2 different sizes of datasets used. It will also go over both the backbone model and go in-depth on the continual learning strategies used for the two different models. Lastly, it will go over the

metrics used to measure the performance of the models.

2.1. Dataset

The dataset is from is called The EgoObjects Dataset which is an egocentric video dataset [4]. Then from there, the videos are cropped to get an image of the object in the dataset. The challenge dataset is too large so I am using the demo dataset that was released earlier for the competition. The demo dataset is divided into two sets. The training dataset has labels and the test dataset does not have labels. The demo dataset is comprised of 90 classes with an imbalance in the number of classes, meaning there are 6 classes per experience. Then from there, I created a smaller dataset of 15 hand-selected classes from the demo dataset so that there is only 1 class per experience. The total number of images and the label distribution are as follows:

Table 1. Label distribution of the two datasets

Dataset	Total Images	Mean	Standard Dev	Median
90 classes	6619	97.338	45.507	76.0
15 classes	1072	76.571	18.084	71.5

Some sample images from the demo dataset

Figure 1. Scarf [2]

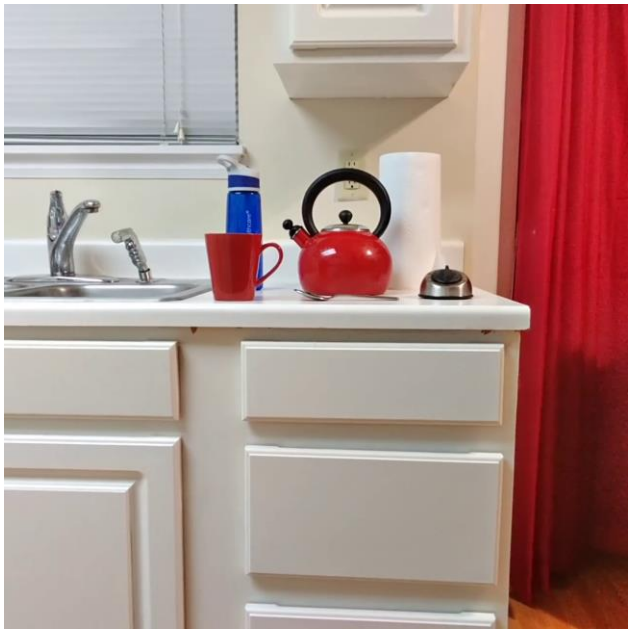

Figure 2. Teapot [2]

Figure 3. Can opener [2]

Other similar datasets that have short video sessions similar to EgoObjects are CORE50[2] and OpenLORIS [3]

2.2. Model

The backbone model is a typical deep learning model before we apply continual learning strategies to help mitigate catastrophic forgetting. The backbone model is used in both the base model and the new model. The base model which includes the backbone model is taken from the 2nd place winners (BaselineCL) for this challenge which uses a RegNet backbone model with a class-balanced replay buffer. The new model is a modification of the base model which adds the plugin Memory Aware Synapses (MAS)

2.2.1 Backbone Model

The backbone model is Regnet_x_16gf which the 2nd place winners found to result in the best performance with the models they have tested. RegNet_x_16gf uses a bottleneck layer with the use of convolution blocks made up of convolution functions, batch norms, and a ReLU activation function [5]. The model is pretrained using ImageNet-1K. The model is trained with a NVIDIA GeForce RTX 3050 4GB Laptop GPU using stochastic gradient descent with 3 training epochs. The initial learning rate is set to 0.2 which is adjusted using cosine annealing. The momentum is set to 0.9 and the weight decay is 1e-5 [6].

2.3.2 Base Model

The base model uses a replay buffer with a class balance storage policy. Replay is known to be one of the most effective strategies for dealing with catastrophic forgetting [1]. The replay uses images from previous experiences to use whenever the model is going to be trained for another experience to make sure it does not forget the classes from the previous experiences. Images are introduced and mixed in a training time along with the new classes that are learned. With the class balance the replay will store an equal number of images per each class in the replay buffer which allows the classes to be equally not forgotten as opposed to the default which is an experience-based buffer that balances the number of images per each experience which may lead to a class imbalance in the replay buffer if there are multiple classes learned in each experience. The challenge rules state that the maximum size of the replay buffer is 3500 which is what the 2nd place winners used for their submission [6]. Since the demo dataset and the smaller dataset has fewer images, I set the replay buffer memory size to 350. One of the reasons is that storing many images in the replay buffer compared to the dataset which results in the last experience basically just training the model on the entire dataset which would make the continual learning process trivial. The other reason is that it could lead to overfitting in some of the classes in the earlier experience since most or all of the class is going to be trained at every experience and every epoch. Hence why reducing the size of the replay buffer is necessary.

2.2.1 New Model

The new model that I will develop is the same as the base model but with the addition of Memory Aware Synapsis (MAS). MAS is a parameter regularization scheme that computes the importance of each parameter in the model calculated by seeing how sensitive the predicted output function is to a change in the parameter [4]. In each experience when learning new classes if an important parameter is going to change then it is penalized [4]. Which in theory is meant to prevent important parameters for previous classes from changing to prevent them from being overwritten. The hyperparameters used for MAS are lambda_reg and alpha. The weighting of the penalty term for the loss function is controlled by lambda_reg which has a default value of 1 [1]. Alpha is used to update the importance of the parameter by also considering the influence of the previous experience and is set to a default of 0.5 [1].

2.3. Metrics

The main metric of the competition is Average Mean Class Accuracy (AMCA). After each experience, the model will be tested on the entire test set. Then the accuracy for each experience is averaged to get the AMCA. The other metric that I will use is the accuracy of the last experience (experience 15). Since by this point all the training data has gone through the model and measuring the accuracy of the test data on the last experience will ideally measure how well the model will generalize on different and new training samples but will not necessarily tell you how well the model will learn a new class which is the point of the field of continual learning.

3. Results

There will be 3 experiments conducted, one being the main experiment where both models are run with both datasets. The second one is the hyperparameter experiment which modifies the MAS hyperparameter that weights the penalty term in the loss. The last one is the ablation experiment where I would be adding a SoftMax layer at the end of the base model so the SoftMax layer is the output, which is standard for classification tasks, but using a sequential to add the SoftMax layer makes the model unable to train and test so there are no results to report. More on this in the discussion section.

3.1. Main Experiment

The main experiment was conducted using both datasets with both models. There seems to be a bug that with the replay plugin with this devkit the epochs won't finish for the training phase which may cause some errors. They have both been measured using AMCA and the accuracy of the model in the last experience. Table 2 with the smaller 15-class dataset.

Table 2. Comparison of the models with the smaller dataset

Model	**AMCA**	**Experience 15**
Base	0.2400	**0.5122**
New Model	**0.2793**	0.4851

Table 3 is where the models are compared with the larger 90-class dataset.

Table 3. Comparison of the models with the smaller dataset

Model	AMCA	Experience 15
Base	**0.3582**	**0.6408**
New Model	0.3455	0.6302

3.2. Hyper-Parameter Experiment

The hyper-parameter that is going to be optimized is lambda_reg for the Memory Aware Synapsis (MAS). It is going to be tested on the new model with the larger demo dataset. The results are ordered in the order I tested them.

Table 4. AMCA and accuracy of experience 15 with different lambda parameters for MAS

Lambda_reg	AMCA	Experience 15
1	0.3456	0.6302
2	0.3179	0.6188
0.5	0.3502	0.6592
0.6	**0.3823**	0.6482
0.7	0.3558	**0.6719**
0.4	0.3109	0.6184

4. Discussion

There is some bug with the devkit environment and the replay buffer that when the images from the replay buffer get added to the training batches then either the epoch never finishes since the replay buffer adds new batches or it says it adds more batches to the training loop but doesn't. If it is the former which I am more inclined to believe that it is then there will be some errors in the results. It is also slightly outdated as this is a growing field and there were a few changes I had to make on the devkit already. Either way, there are some notable results in the Main Experiment and Hyper-Parameter Experiment. As well as the reasoning as to why I can not do the Ablation Experiment.

4.1. Main Experiment Discussion

Both models did better on the larger dataset than on the smaller dataset. This could come from the bug in which the replay buffer has significantly more images per experience than any single class, but this also could be caused by the fact that there are going to be more images in the replay buffer than there are in each experience so the replay buffer is going to have more influence in modifying the parameters than the single class does in each experience. It could also be in part to that since the replay buffer will probably have more influence in changing the model parameters which may result in a generalization error since some of the images are going to be trained repeatedly for each experience. Another notable result is that the baseline model outperformed the new model in both accuracy metrics for the 90-class dataset. This could be due to the fact the new model is using the default parameters for MAS which may give too much penalty for changing the weights of the model.

4.2. Hyper-Parameter Experiment

For the hyper-parameter experiment, I chose to optimize the lambda_reg parameter for the MAS continual learning strategy. The reason is that the MAS plugin is the only difference between the base model which was taken from the second-place winner which means the MAS plugin was the only thing not optimized. The reason I choose to optimize lambda instead of the alpha hyperparameter is that I have more of an intuitive understanding of how the lambda hyperparameter affects the model compared to the alpha parameter. For the range of values tested for the lambda hyperparameter, I decided to try the range of ½ to 2 times the default value which is 1. As Table 4 shows lambda being 0.5 performed better than lambda being 1 from there I tested values around 0.5 with a step size of 0.1 to try and find the optimized value. If we are going by AMCA which is the metric that the challenge competition uses then lambda being 0.6 performs best, but if we are going by how well the model would perform in practice then lambda being 0.7 is best since the accuracy on experience 15 is when the model is finished all the training for all the classes and would be used to classify any new images.

4.3. Ablation Experiment

What I wanted to do is use a SoftMax layer as the output since SoftMax is typically used for multi-class classification tasks. The output layer of the RegNet_x_16gf is a linear layer so in theory, it should perform better. I did not want to use a new layer in the beginning or in the middle of the model since that may mess with the pretrained weights and may end up performing worse. I tried to add the layer by putting both the RegNet model and the SoftMax layer in a sequential model, but then the model would not train and I am unsure of how to fix that. It could also be the case that the devkit does not work with a full model inside of a sequential model, but I am not sure. In any case, I cannot run the model with the SoftMax layer. I would not expect the model to perform better by adding any layer because the training hyperparameters rely on the fact the model is pretrained and adding any new layer or removing a layer would result in the model having to retrain instead of having transfer learning do most of the work. If there was going to be any architecture change it would have to be with a whole new model that is pretrained otherwise it

would be too slow to train properly especially given the time constraints of the challenge task.

5. Conclusion

The field of continual learning is to mitigate catastrophic forgetting when new classes are learned by the model. For this challenge track, 15 separate training experiences added classes incrementally. The two datasets that I used were both smaller versions of the challenge dataset. One with 90 classes and 6 classes per experience and one with 15 classes with 1 class per experience. The baseline model was from the 2nd place winner of this challenge track which used a class-balanced replay strategy. From there I made a new model that used Memory Aware Synapsis to try and stop important parameters in the model from being overwritten by penalizing the loss function if it did so. The metrics used to measure the performance of the models were the average mean class accuracy and the accuracy of the last experience. With unoptimized Memory Aware Synapsis hyperparameters the baseline model outperformed the new model, but with the lambda hyperparameter optimized it outperformed the baseline model in both metrics.

References

[1] Lomonaco, V., Pellegrini, L., Cossu, A., Carta, A., Graffieti, G., Hayes, T. L., ... & Maltoni, D. (2021). Avalanche: an end-to-end library for continual learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 3600-3610).

[2] Lomonaco, V., & Maltoni, D. (2017, October). Core50: a new dataset and benchmark for continuous object recognition. In Conference on Robot Learning (pp. 17-26). PMLR. https://vlomonaco.github.io/core50/

[3] She, Q., Feng, F., Hao, X., Yang, Q., Lan, C., Lomonaco, V., ... & Chan, R. H. (2020, May). Openloris-object: A robotic vision dataset and benchmark for lifelong deep learning. In 2020 IEEE international conference on robotics and automation (ICRA) (pp. 4767-4773). IEEE. https://lifelong-robotic-vision.github.io/dataset/object

[4] Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., & Tuytelaars, T. (2018). Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 139-154).

[5] Xu, Jing, Yu Pan, Xinglin Pan, Steven Hoi, Zhang Yi, and Zenglin Xu. "RegNet: self-regulated network for image classification." *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[6] Xu, T., Feng, H., Zheng, S., Yang, X., Liu, Z. (2022). The 2nd Place Solution for CVPR 2022 Workshop on Continual Learning (CLVision, 3rd Edition) Challenge–Track 1: A Replay-based Continual Learning Approach