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

Anonymous CVPR submission

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. For the first challenge track of the CVPR CLVISION workshop is focused on class incremental learning which is learning new classes in separate phases. In each phase only the samples of the new classes can be used, and the samples of previously used data is unavailable. For this challenge track there is a template provided with several commonly used strategies for continual learning. Thus, the purpose of this project is to test how effective each strategy is using the data set provided.

1. Introduction

The first challenge track for CLVISION is an instance classification track. In this track the focus on how to mitigate catastrophic forgetting while going through a series of learning different classifications. The challenge tracks also comes with a devkit that is provided that also contains commonly used continual learning strategies that are readily available as a plugin [1]. These plugins are from a library called Avalanche which is an End-to-End Continual Learning Library Based on PyTorch [1]. So, for my project the goal will be to find the best strategy or strategies available for the dataset that is provided. Finding strategies to mitigate catastrophic forgetting will allow models to be able to run and learn in more dynamic 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 in continual learning will be beneficial for many areas both research and real-world applications.

1.1. Instance Classification Challenge Track.

The Continual Instance Classification challenge track is of course a classification task. The challenge is structured as a Class Incremental model in which there are multiple phases of training and testing. There will be 15 of these experiences and a total of 1100 categories which is about 74 categories per experience. The inputs are a 224x224 RBG images and the outputs is the class. This is a fully supervised task so there are labels provided at training time. To measure the accuracy of the outputs we will be using the average mean class accuracy which is for each pass on the test step the mean class accuracy is calculated and then the 15 results are averaged to get the final result.

1.2. Dataset

The dataset is from is called The EgoObjects Dataset which is a video dataset which is egocentric [4]. Then from there the videos are cropped to get an image of the object in the dataset.

1.3. Sample Images

Figure 1. Scarf [4]

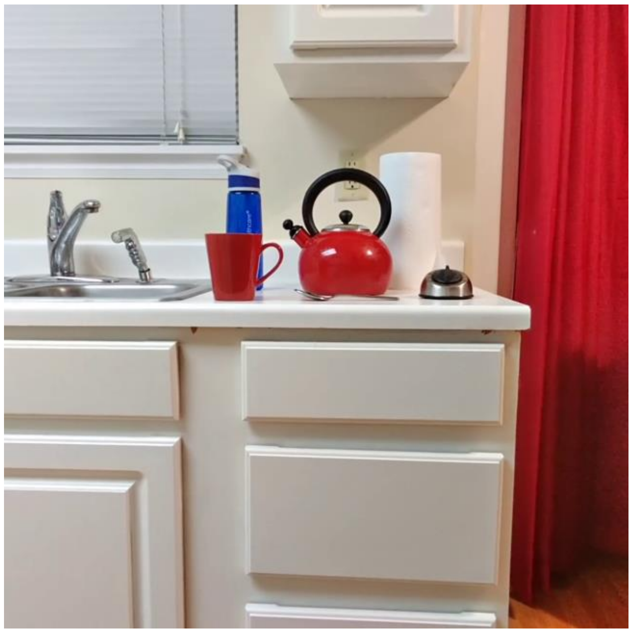

Figure 2. Teapot [4]

Figure 3. Can opener [4]

.

1.4. Project feasibility and required resources

Since most of the code is already written most of the project will be about understanding the different learning strategies and actually running the code. In which case the only hurdle is how long will the training runs take and how many. There are 15 different strategies provided with avalanche so I would like to at least be able to test each and see how accurate each one is and if time permits try using hybrid models with the strategies that performed the best. Other submissions [3] used 1 GPU so I believe that 1 GPU is all that I would require. There is a demo set that is significantly smaller than the actual set which will help me finetune the model.

2. Instance Classification Competition

The instance classification track had 37 different participants the score was based on the average correct classification from stream of 15 experiences. The scores of from 20 to 2 were spread out across the range of 0.5004 to 0.5232 while the number 1 score was 0.5635. The competition is currently archived.

2.1. Code and Strategies

The is no code that I could find from the top 3 finishers from the Instance Classifications track, and I have searched for code on this challenge track for hours without any success. So instead, I will go over getting the dev kit set up as well as the strategies from the second-place finisher. Which did some testing with different model and strategies and took the one with the best result

2.1.1. Setting up Devkit

You can download the devkit from the challenge website or from the GitHub `https://github.com/ContinualAI/clvision-challenge-2022`. There were 2 modifications required to get the code running. The first one is changing how it created a directory. The first of which datetime.datetime.now().isoformat(). The “:” needed to be replaced by “-“ because windows does not allow directories to be named with the character “:”. The second change I had to make is that torchvision.datasets.Kenetics400 is depreciated in favour of torchvision.datasets.Kenetics and then when you instantiate the class you have to specify in the constructor that num_classes needs to be equal to “400”. After these modifications were made there is a starting template that runs a Naïve strategy which just fine tuning strategy. Which worked, but the Naïve strategy is not good because it is very prone to catastrophic forgetting because it does not regulate the change of weights nor stores samples to call back later.

2.1.2 2nd place BaselineCL

They tested with different backbone. Mainly Resnet, Efficientnet and Regnet in which Regnet preformed the best with batch normalization and label smoothing [3]. Batch normalization is to accelerate the rate of convergence and to make the training process more stable [3]. Then for continual learning strategies they choose the replay strategy which stores a small number of samples so when it is training for a new tasks the samples are run through with the new task data to remember the old data. They also tested and added class balancing which paired with replay gave the accuracy result of 0.4961 compared to 0.4939 with just the replay strategy [3].

3. Research Papers

A couple of research papers that are related to the project. That will help create an effective model for the classification task.

3.1 Avalanche: an End-to-End Library for Continual Learning

Avalanche is the primary library of the continual learning community and the library that is used in the devkit. While it goes through all the library categories: Benchmarks, Scenarios, Strategies, Metrics and Loggers the focus I want to look at is the strategies because that will be the only thing, I will have to implement for the challenge track. Instead of manually calling the training, evaluation, and testing method from a strategy. With Avalanche each continual learning algorithm will have two modes. The first of which is a training mode which is used to update the model and evaluation mode which can be used to process streams of experience for testing [1]. All that is required is adding the strategy to a list of plugins [1]. Which makes making hybrid models with multiple strategies very simple as you just need to add multiple strategies to a list of plugins. Another thing this plugin strategy accomplishes is to increase reproducibility because there is no worry with people coming up with different representations for the same strategy [1]. Plugins are not exclusive to strategies you can also use plugins for logging data and measuring metrics. In which case it makes it easy to use and create models using Avalanche.

Future works:

- Adding additional learning paradigms
    - Reinforcement learning
    - Unsupervised Learning
- Adding task types:
    - Detection
    - Segmentation
- Application Contexts:
    - Natural language processing
    - Speech Recognition

3.2. Simpler is Better: off-the-shelf Continual Learning through Pretrained Backbones

Using pretrained backbone models we can achieve strong results for most common benchmark metrics for a model [2]. This is done by reordering data in the training phase and exploiting the power of pretrained models to computer a class prototype and fill a memory bank and from there we match the closest prototype through a KNN-like approach which gives us a prediction [2]. The overall memory cost of less than 10 KiB for each of the backbones model tested. Also that Visual Transformers tended to generalized better with respect to CNN models or at least, more discriminative features [2].

Future Works:

- Substituting the knn approach with linear classifier
- Extend the use to unsupervised learning

.

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] Pelosin, F. (2022). Simpler is better: off-the-shelf continual learning through pretrained backbones. *arXiv preprint arXiv:2205.01586*.

[3] Xu, T., Feng, H., 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.

[4] Pellegrini, L., Zhu, C., Xiao, F., Yan, Z., Carta, A., De Lange, M., ... & Vazquez, D. (2022). 3rd Continual Learning Workshop Challenge on Egocentric Category and Instance Level Object Understanding. *arXiv preprint arXiv:2212.06833*.