# ICCV 2021 Workshop SSLAD Track 3A - Continual Object Classification 3rd Place Solution

Azamat Sabyrbayev
BTS Digital
Nur-Sultan
azamat.sabyrbayev@btsdigital.kz

## Abstract

*Continual Learning (CL) nowadays is actively being applied in different areas of Computer Vision such as autonomous driving, retail, etc. The main goal of this task is to handle the problem of catastrophic forgetting, where a new learned model suffers from recognizing older learned data. This paper proposes a collection of strategies which helps to train a robust classification CL model for autonomous driving on SODA10M large-scale dataset. Our approach firstly uses a memory population approach to tackle class imbalance and catastrophic forgetting problems. The memory population approach takes into account real world class distributions and upsamples to the equal number of objects. Additionally we use a custom learning rate scheduler which reduces the learning rate after each experience. These strategies helped to get 65.54% average mean class accuracy (AMCA) on the Test set.*

## 1. Introduction

During the past decade deep convolutional neural networks have become a crucial method to solve and automate multiple real world problems. This includes classification, detection, generation(GANs), etc. Mostly, in everyday life researchers and engineers solve aforementioned tasks by training offline and serving the prepared model for production.

However, these tasks become more difficult when the data is imbalanced and comes from the stream. This concept is called Continual Learning (CL). CL is an idea to train a model for a wide range of tasks sequentially without having access to the previously used data. During the learning process the model has the knowledge obtained from the preceding tasks. This concept is more closely related to how a human learns in daily life.

### 1.1. Challenge

This paper describes the solution of a 3rd place winner on SSLAD 2021 Continual Object Classification challenge. The main goal is to classify 22249 chronologically ordered object centered images in 6 categories (pedestrian, cyclist, car truck, tram, tricycle). Moreover, a specific requirement was to use Avalanche (`https://avalanche.continualai.org`) library, which is a specifically designed framework for CL tasks. The framework contains a good set of implemented techniques/strategies for CL. There were specific rules for the classification track (3A):
1. ImageNet pre-trained models can be used, but those with no more than 25M parameters
2. No upsampling allowed. However, it is allowed to upsample from old data in a memory buffer
3. Maximum batch size is 10

### 1.2. Dataset

Only labeled part of the SODA10M dataset was used in training and testing phases during the competition. The dataset consists of 6 classes – pedestrian, cyclist, car, truck, tram, tricycle. Distribution of classes over experiences is presented in Table 1. Obviously the most challenging class was Tricycle. In all experiences the number of samples of this class is the lowest, which caused participants enough problems during the challenge. At the end of the competition the maximum score for class "tricycle" among all participants in test set was only 25.78%.

### 1.3. Evaluation Metric

The leaderboard was evaluated with average mean class accuracy (AMCA). At 6 points during the training stream, the model was evaluated on the test/validation set. At each evaluation point, the mean class accuracy is calculated. Then, these 6 results were averaged to get the final result.

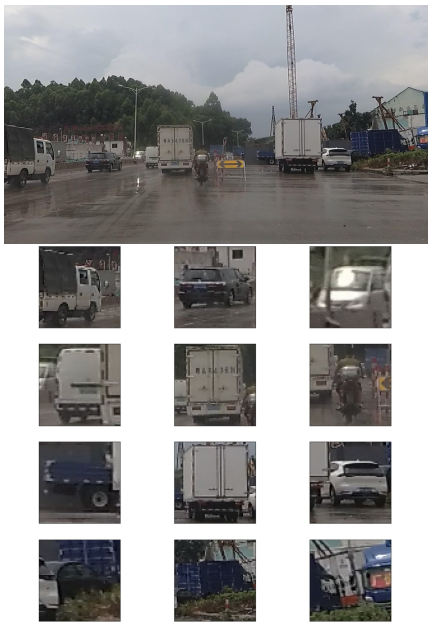

Figure 1. Top: sample image from the dataset. Bottom: images cropped from the image and resized to size 64x64 to further feed into the network

| Experience | Pedestrian | Cyclist | Car | Truck | Tram | Tricycle |
|---|---|---|---|---|---|---|
| Experience-0 | 300 | 821 | 2498 | 1324 | 157 | 57 |
| Experience-1 | 0 | 0 | 721 | 400 | 33 | 0 |
| Experience-2 | 972 | 442 | 3066 | 2051 | 190 | 21 |
| Experience-3 | 0 | 0 | 1722 | 744 | 94 | 0 |
| Experience-4 | 259 | 55 | 2939 | 895 | 367 | 2 |
| Experience-5 | 303 | 397 | 1235 | 77 | 106 | 1 |

Table 1. Class distribution over experiences

## 2. Methodology

### 2.1. Replay Buffer

The main challenge of this competition is highly imbalanced data. As we can see from Table 1, pedestrian, cyclist, tram and tricycle classes are very rarely presented. Also some classes do not exist in some experiences, which leads to the catastrophic forgetting problem. So to tackle these problems we used a replay buffer. The size of the buffer is 1000. The buffer works in the following manner:

1. After experience-0 we sample classes from the input stream with predefined weights and save samples in the buffer. Weights for sampling from the input stream to the buffer are presented in Table 2. As you can see from sampling weights from Table 2, we try to keep all samples of tricycle class in the buffer until the end.

2. Before starting experience-1 we create 2 data loaders: 1 data loader of input stream with batch size 4, and second data loader from buffered data with batch size 6. Lengths of 2 data loaders are the same. We sample samples from the buffer data loader with uniform distribution between samples during the training. Because the size of buffered data is only 1000 we use upsampling methods to align the buffer data loader's size to the input stream's data loader's size.

| Classes | Pedestrian | Cyclist | Car | Truck | Tram | Tricycle |
|---|---|---|---|---|---|---|
| Weights | 2 | 10 | 1 | 1 | 10 | 10000 |

Table 2. Weights for sampling from input stream into buffer

3. Before starting experience-2 we divide buffer by 2 and save in the first half samples from experience-1, and in the other half we keep a part of samples from experience-0, which already was in buffer, and drop a part of them from buffer. Weights for keeping and dropping samples from the buffer are the same as presented in Table 2. Further, we repeat the algorithm equally dividing buffer space between experiences.

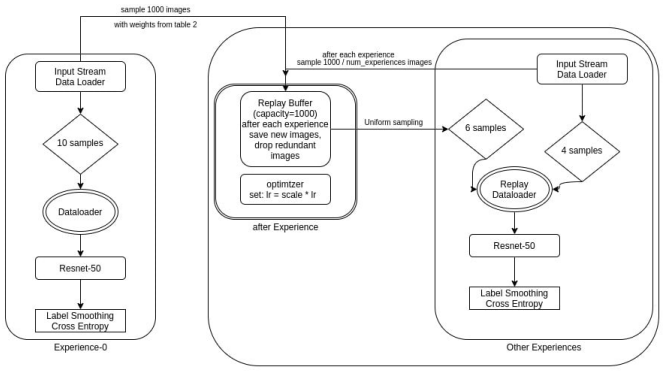


Figure 2. Training pipeline

### 2.2. Architecture

By competition rules model was restricted to have parameters less than 25 million. We used ResNet-50 architecture initialized with weights pre-trained on ImageNet, which has about 23 million parameters. Also DenseNet169 and EfficientNetV2-small had good performances, but we stick on ResNet-50 because experimentally it performed better than aforementioned models.

### 2.3. Loss Function

Our loss function (label smoothing cross-entropy) for a sample is defined as follows: where l is loss, sc - smoothing

$$l = -(1 - sc) \sum_{i=0}^{N} y_i * \log(p_i) - sc \sum_{i=0}^{N} \log(p_i)$$

coefficient, N - number of classes, $y_i$] - label of class i, $p_i$ -

probability of class i predicted by a model Loss L of a batch of size B is calculated as follows:

$$L = \sum_{i=0}^{B} l_i$$

### 2.4. Augmentations

We used horizontal flip, perspective transform and random erasing transforms with probability of applying 0.5 (Figure 3). During the training process we've also tried more aggressive augmentations, but it was fruitless.

Figure 3. Augmented images from Figure 1

### 2.5. Training

We used Adam optimizer with initial learning rate 2e-4, after every experience learning rate was updated by multi step scheduling. MultiStep scheduler multiplies learning rate with factor 0.5 after first experience, and after other experiences multiplying factor is set to 0.7. Smoothing coefficient (sc) of cross-entropy loss is set to 0.1. Batch size was fixed and equal to 10. During the first experience, batch size consists of only input stream's samples and no samples from the buffer are used, because it is empty. After the first experience, a batch is formed from 4 samples from the input stream and 6 samples from the buffer (Figure 2). Each experience trained only one epoch, the iteration number is equal to the input stream's data loader's size, so input stream samples are seen to the model only once. Also, there are no multiple iterations over one batch. Also, input stream is not shuffled, that is, it goes in chronological order. Input image size is set to 64. Training was done on the TITAN RTX device. RAM used while training is about 1.5-2 Gb.

## 3. Experimental Results

Table 3 shows the accuracy of each class over experiences. It is obvious that rare occasions of a class in an experience leads to the performance degradation. For example, pedestrian's accuracy degrades notably from experience-0 to experience-1, because there is no pedestrian class in experience-1. On the other hand, car and truck classes are often present over all experiences and thus they have good performance. So we can see that our approach solves the catastrophic forgetting problem only partially. Also, we can noticed that after experience-0, rare presented classes' accuracy starts improving. For example, the tricycle class is not presented in experience-1, but the accuracy improves because of using buffer and upsampling the class in experience-1. Without buffer the accuracy will stick on 0.

| Experience | Pedestrian | Cyclist | Car | Truck | Tram | Tricycle | Mean |
|---|---|---|---|---|---|---|---|
| Experience-0 | 82.1% | 38.0% | 78.1% | 91% | 19.5% | 0% | 51.5% |
| Experience-1 | 52.2% | 65.4% | 93.0% | 79.6% | 51.3% | 17.0% | 59.7% |
| Experience-2 | 79.1% | 72.5% | 95.2% | 86.0% | 70.1% | 13.6% | 69.9% |
| Experience-3 | 79.9% | 70.1% | 91.3% | 93.4% | 61.3% | 34.4% | 71.8% |
| Experience-4 | 75.8% | 69.3% | 95.5% | 92.8% | 87.7% | 33.5% | 75.8% |
| Experience-5 | 79.9% | 82.0% | 95.5% | 90.2% | 86.6% | 27.7% | 77.0% |

Table 3. Accuracy of each class over experiences on the validation set

Overall, accuracy on the validation set is 67.6% and on the test set is 65.5%

| Experience\ Method | Best Model ResNet50 | ResNet50+CWR star | EfficientNetV2-small | ResNet50+ learning w/o forgetting plugin |
|---|---|---|---|---|
| **Experience-0** | **51.5%** | 48.9% | 47.2% | 50.2% |
| **Experience-1** | 59.7% | **60.1%** | 48.9% | 58.1% |
| **Experience-2** | **69.9%** | 66.9% | 66.5% | 69.2% |
| **Experience-3** | **71.8%** | 66.1% | 62.9% | 71.0% |
| **Experience-4** | **75.8%** | 68.8% | 71.0% | 71.3% |
| **Experience-5** | **77.0%** | 72.4% | 71.9% | 74.5% |
| **Mean** | **67.6%** | 63.8% | 61.5% | 65.7% |

Table 4. Mean accuracy by experience for different methods

During the competition other experiments were also tested. However the results were quite low. Some of them are presented in Table 4. For example:
1. Copy Weight with Reinit (CWR star). Shortly, this method straightly copies the weights from the previous task and further works with them.
2. EfficientNetV2-small model is one of the latest SOTA models. This method also didn't give a good score. Probably, it only works on high image sizes.
3. Multiple iterations over a batch didn't work, 2 successive

iterations gives the same result as one iteration, more iterations makes performance worse.
4. Mixup augmentation also didn't help, even if we iterate over one batch multiple times.
5. Applying learning w/o forgetting plugin showed a good result, but not the best. LwF uses distillation to regularize the current loss with soft targets taken from a previous version of the model.

## 4. Conclusion

In this work, we examined the issue of online continual learning from severely imbalanced, temporally correlated streams from different domains. Moreover, we proposed a memory population approach to tackle problems in such learning settings. Also we provided augmentations and learning rate scheduling methods to improve the model's performance. As we can see from the results, class imbalance and catastrophic forgetting problems were solved only partially. As further improvement we can use the interpolation method before the model's first layer to resize input image from 64 to bigger sizes, because ResNet-50 and EfficientNetV2-small models were pretrained on images with bigger image sizes. Also center crop augmentation may help to improve the score, because in some images there are parts of objects from other classes, so it can help to get rid of them.