Subset Replay based Continual Learning for Scalable Improvement of Autonomous Systems

Pratik Prabhanjan Brahma
Electronics Research Laboratory
Volkswagen Group of America
pratik.brahma@vw.com

Adrienne Othon
Electronics Research Laboratory
Volkswagen Group of America
adrienne.othon@vw.com

Abstract

While machine learning techniques have come a long way in showing astounding performance on various vision problems, the conventional way of training is not applicable for learning from a sequence of new data or tasks. For most real life applications like perception for autonomous vehicles, multiple stages of data collection are necessary to improve the performance of machine learning models over time. The newer observations may have a different distribution than the older ones and thus a simply fine-tuned model often overfits while forgetting the knowledge from past experiences. Recently, few lifelong or continual learning approaches have shown promising results towards overcoming this problem of catastrophic forgetting. In this work, we show that carefully choosing a small subset of the older data with the objective of promoting representativeness and diversity can also help in learning continuously. For large scale cloud based training, this can help in significantly reducing the amount of storage required along with lessening the computation and time for each retraining session.

1. Introduction

Recent advances in machine learning have shown tremendous improvements in solving various computer vision problems. Deep learning has stood out to be the state-of-the-art for object classification [15], activity recognition [3] [23], semantic segmentation [25] [20] and depth estimation [18] [4]. Machine learning, and especially deep learning, is thus expected to massively influence the production process of perception systems for autonomous vehicles of the present and the future. However, when these models are fine-tuned with new data, a substantial degradation is seen on the performance with the original data. This process is known as catastrophic forgetting [22].

One has to store and repeatedly train with all the data simultaneously in order to ensure reliable performance on both newer and older scenarios. This sort of joint training causes a lot of scalability issues in regards to storage and computation. Data collected through driving can often be huge given that the output of the various sensors like cameras, lidar and radar as well as internal signals from CAN bus and other measurement units can easily be of the order of hundreds of megabytes per second. The observations are also not equally informative. It is difficult to come up with either a one-time dataset or modeling that can cover all possible driving situations that an autonomous vehicle may encounter. Driving rules, traffic behavior and weather conditions change from one place to another and over time. So, it will be necessary to continuously collect such new instances as they are encountered in order for the existing systems to adapt accordingly. Thus, there is a need for the perception modules to learn continuously and reliably from new data in a scalable manner but without compromising on the knowledge obtained from previous training. Figure 1 presents our proposed flow of continuously improving the autonomous and advanced driver assistance (ADAS) systems over time. Once a session of training is complete, a carefully chosen subset of the training data belonging to the current session is added to the common memory. The testing phase comprises of a data collection campaign that records novel or edge case scenarios that is supposedly not seen during previous training. This new data along with the replay of the subset memory is used during the training of future sessions.

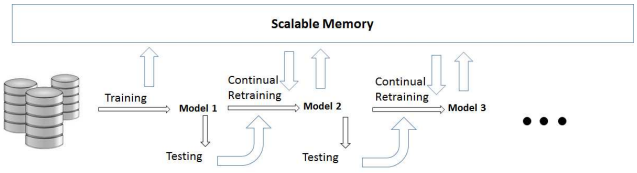


Figure 1. Overview of continual learning for autonomous driving modules using a scalable subset memory on top of sequentially arriving new data

Humans perform a good job in learning of new tasks over their entire lifetime while maintaining a decent performance on already known tasks. During the past few years, several algorithms have been suggested to meet the demands of lifelong or continual learning for typical machine learning models like neural nets. The problem of catastrophic forgetting was discussed in further details in [9] and dropout training was proposed to cure it to some extent. In [13], the authors computed the diagonal of the Fisher information matrix as an indicator of the importance of each particular weight parameter corresponding to a given task and used that to form a Elastic Weight Consolidation (EWC) regularization loss. This is inspired from the neuroscientific hypothesis of synaptic consolidation whereby knowledge about how to perform a previously seen task is encoded in a set of neural synapses which are rendered less plastic and therefore are more stable over time. On the other hand, less-forgetting learning (LFL) [12] applies a regularization condition to make the intermediate layer features of the retrained neural network closer to that extracted by the older network. For supervised classification problems, Learning without Forgetting (LWF) [16] is a technique to add knowledge distillation loss to the overall loss function as a way to preserve the knowledge regarding the previous state of the neural network. There have also been some recent work on memorization and replay techniques for preserving continuity in deep learning models. For example, Gradient Episodic Memory (GEM)[21] used the gradients with respect to randomly memorized subsets of exemplars from previous tasks to calculate the appropriate gradient update while learning continually. Generative replay (GR) [26] tried to teach generative adversarial networks or GANs [8] to learn and generate examples corresponding to each previous task. On a similar note, [29] used autoencoders to do lifelong learning.

An important aspect of all the replay techniques is that the replayed memory is assumed to be well representative of the data belonging to that particular task or data distribution. Choosing the right subset has been a critical function for other associated problems like document or video summarization [7] and active learning. For example, [2] chose the samples from a pool of unlabeled dataset that correspond to highest entropy values on a model's prediction. Submodular sampling on speech data was shown in [30] to outperform random as well as entropy sampling. Similarly, [11] showed the application of submodular sampling for video summarization. In the context of incremental learning for specifically learning from data belonging to new classes, [24] had proposed a *herding* approach to choose a subset of exemplars that causes the average feature vector over all selected samples to best approximate the average feature vector over all training examples. In this paper, we present a general framework for designing submodular functions to select representative as well as diverse subsets of training data. We show how this can help mitigate the problem of catastrophic forgetting without being bottle-necked by storage and computation constraints. Following the introduction, Section 2 illustrates the problem set up for continual learning. In Section 3, submodularity is shown as a general framework for subset selection and various examples of functions are laid out. Experimental results and comparison with some of the contemporary continual learning algorithms are presented in Section 4. Finally, Section 5 provides the conclusion of our work.

2. Problem Formulation

Unlike the conventional setting of supervised training, continual learning attempts at learning when data is fetched in sequential chunks enumerated by sessions. In a supervised setting for the i^{th} session, the objective of a conventional machine learning model is to learn a mapping f_i, parametrized by Θ_i, from the input $\mathbf{X}_i$ to $\mathbf{Y}_i$. The set of classes or categories contained in each session can be different from each other. The total number of unique classes seen across all sessions is given by C. Let us define $\mathbf{X}_{i\rightarrow j}$ to be the concatenation of all training data from session i to j in chronological sequence where $i < j$. We also have similar notations for $\mathbf{Y}_{i\rightarrow j}$ and $\Theta_{i\rightarrow j}$ to denote a list of concatenated output labels and weights respectively. If each session is treated independently, f_i is taught by minimizing a loss function as

$$\Theta_i^* = argmin_{\Theta_i} \ell(\Theta_i, \mathbf{X}_i, \mathbf{Y}_i). \tag{1}$$

The loss is estimated by empirical risk minimization (ERM) where one tries to minimize $\frac{1}{|\mathbf{X}_i|}\sum_{x_a,y_a\in\mathbf{X}_i,\mathbf{Y}_i} \ell(\Theta, x_a, y_a)$. This holds good under the assumption where (x_a, y_a) is treated as an independent and identically distributed (IID) sample from the entire data distribution. However, this doesn't hold good in the case where data is observed in a sequence of sessions with varying distributions.

In case of classification problems, ℓ can be the categorical cross-entropy (CCE) whereas it can be the mean squared error (MSE) for regression problems. For learning in a continuous fashion, we would ideally like Θ_i^* ,that is obtained right after being trained with data from the i^{th} session, to also retain the knowledge obtained from observing the training data $\mathbf{X}_{1\rightarrow i-1}$. Joint training is done when Θ_i^* is taught from training with $\mathbf{X}_{1\rightarrow i}$ and $\mathbf{Y}_{1\rightarrow i}$ together. It can be considered to be the upper bound for all approaches. Continual learning methods like EWC, LWF and LFL assume that they have no access to the previous training data $\mathbf{X}_{1\rightarrow i-1}$ and thus apply a regularization to the loss function while training on the current session

$$\Theta_i^* = argmin_{\Theta_i} \ell(\Theta_i, \mathbf{X}_i, \mathbf{Y}_i) + \lambda \mathcal{R}(\Theta_i, \Theta_{1 \to i-1}, \mathbf{X}_i). \quad (2)$$

In the context of EWC, the model weights $\Theta_{1 \to i-1}$ along with their corresponding Fisher Information coefficients are stored for each session. LWF stores the final layer softmax activations of the current session's input data by doing a feed forward pass through the previously trained model. Methods like GEM memorize a randomly chosen subset to compute the gradient update during new sessions whereas GR stores the weights of session-specific GAN neural nets. Let $\mathcal{M}_i$ denote the maximum hard disk or cloud memory allocated for each i^{th} session. If we plan on only storing a subset of size b from the session's training data, then it should fall well within the $\mathcal{M}_i$ memory constraint. The objective of subset memory selection is to retain the accuracy on the scenarios belonging to the past training when a model is retrained on the new session. Since we do not know what type of data shall be encountered in the future sessions, we cannot use ($\mathbf{X}_i$, $\mathbf{Y}_i$) to choose an optimal subset for sessions 1 to $i-1$. The next section details certain strategies to solve for the subset selection in a robust and cost effective manner.

3. Subset Selection Strategy

3.1. Submodularity

Submodularity is a property of set functions. A set function $f : 2^V \to \mathbb{R}$ returns a real value $f(S)$ for any subset $S \subseteq V$. A function f is submodular [14] if for every subset A and B in V, we have

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B). \quad (3)$$

An equivalent definition is that given arbitrary sets T and U such that $T \subseteq U \subseteq V$ and element e which is present in V but not in T or U, a set function f is considered to be submodular if it satisfies the diminishing returns property

$$f(T \cup \{e\}) - f(T) \geq f(U \cup \{e\}) - f(U). \quad (4)$$

The latter definition just means that the advantage gained by adding an item $\{e\}$ to a bigger set U is less as compared to adding the same item to T which is a subset of U.

Maximizing submodular functions is an NP-hard problem. But a major advantage of rephrasing the optimization problem of subset selection as a submodular function maximization is that powerful guarantees exist for linear time approximation algorithms. For example, even a simple greedy algorithm can solve the problem with a worst case approximation factor of $(1-e^{-1})$ [14]. It is usually even better than that. This is very attractive because the quality of the solution does not depend on the size of the dataset and hence these algorithms are often scalable to bigger datasets.

3.2. Submodular functions for subset selection

Various clustering algorithms, like k-medoids, can be connected to submodular optimization. For example, the aim of k-medoids is to minimize the sum of pairwise distances or dissimilarities $d(.,.)$ between cluster centers and elements of the entire dataset V. The k-medoids loss is defined as

$$L(S) = \frac{1}{|V|} \sum_{e \in V} min_{v \in S} d(e, v) \quad (5)$$

where S is the set containing all the medoids. This can be converted into a submodular optimization problem [6] by introducing an exemplar element e_0 as

$$f(S) = L(e_0) - L(S \cup \{e_0\}). \quad (6)$$

Maximizing the now submodular function above is the same as optimizing the k-medoids loss. A similar yet popular example of submodular functions is called the facility location problem. Given a set of locations $V = \{1, 2, ..N\}$, the objective is to open facilities in some of them in order to serve a collection of m customers. Opening a facility at location j provides a value denoted by $M_{i,j}$ to customer i. If each customer i is served by the nearest facility which is given by the facility with maximum value, then the total value given to all customers by a subset S of locations is

$$f(S) = \sum_{i=1}^{M} max_{j \in S} M_{i,j}. \quad (7)$$

If $M_{i,j} \geq 0$, then the function is monotone submodular. Other examples of submodular functions can also be entropy $f(S) = -\sum P(X_S) log_2(P(X_S))$ and similarly mutual information.

A computational constraint in the pairwise similarity (or distance) graph based submodular formulations mentioned above is that one needs to calculate and store an $O(N^2)$ matrix where N is the number of training examples. This becomes tough when N is in the order of millions or more. Submodular functions can also be defined over representational features of the given input. A feature based submodular function can be given as

$$f(S) = \sum_{u \in U} g(m_u(S)) \quad (8)$$

where $g(.)$ is a non-negative monotone non-decreasing concave function, u is a particular feature in a set of features U and $m_u(S) = \sum_{j \in S} m_u(j)$ is its non-negative score for a particular feature u over a given subset S. As proved in [28], the sum of concave functions over modular functions (functions that hold the equality in (1)) is a submodular function itself. In [30], the authors used this for large scale speech text subset selection. They used term frequency

inverse document frequency (TF-IDF) value as their non-negative score for each training datum. Over multiple iterations, a greedy algorithm tries to scan over the entire set of features and pick that particular element e which maximizes the conditional gain from its current submodular value. For every iteration t,

$$e_t^* = argmax_{e_t \in V - S_{t-1}}(f(S_{t-1} \cup e) - f(S_{t-1})). \quad (9)$$

While dealing with images and videos, deep learning has been largely successful in coming up with feature extractors which perform way better for learning tasks as compared to hand crafted features. The intermediate layer output of a deep neural network with rectified linear unit (ReLU) activation can be approximated as a non-negative score substitute that also contains class conditional information. Let's assume that we are extracting features from the l^{th} layer of a neural network that has d neurons with ReLU activation. For every input data X^j, the intermediate feature output h^j is a $d-$dimensional vector containing only non-zero entities. For a dataset containing N observations, we have a feature matrix $\mathbf{H} \in \mathbb{R}^{N \times d}$. Let a_k denote the fraction of times when the neuron k is activated (or nonzero) in the feature matrix. Similar to the computation of TF-IDF, we calculate $i_k = |log(\frac{1}{(a_k+\delta)})|$ where δ is a very small number to avoid division by zero in cases where a particular neuron is never activated. Here, i_k is an indication of how common a neural activation is. When multiplied with the actual feature output, it tries to diminishes the weight on commonly occurring neural activations as compared to increasing the importance of rarely occurring ones. For every data point i, we normalize the actual feature value h_k^i for every neuron by multiplying it with i_k as

$$t_k^i = h_k^i \times i_k \quad (10)$$

For every feature or neural activation k, we then compute the modular score for a given subset S as

$$m_k(S) = \sum_{i \in S} t_k^i. \quad (11)$$

One can also come up with other designs for the scoring function as long as the non-negative modularity is maintained. The square root function is used as the concave non-decreasing function $g(.)$ and iterative greedy selection is used to fill the subset till the budget is reached. The overall submodular function is

$$f(S) = \sum_{k=1}^{d} \sqrt{m_k(S)}. \quad (12)$$

3.3. Coverage with Diversity

Maximizing any of the aforementioned submodular functions helps in promoting coverage or relevance of the subset with respect to the whole dataset. In addition to that, a diversity reward can also be included in order to achieve less redundancy among the samples chosen. This has been successfully used in video summarization [11] where the focus is to come up with a succinct yet informative summary of otherwise long videos. One such diversity reward function [17] can be designed as

$$D(S) = \sum_{j=1}^{K} \sqrt{|S \cap P_j|} \quad (13)$$

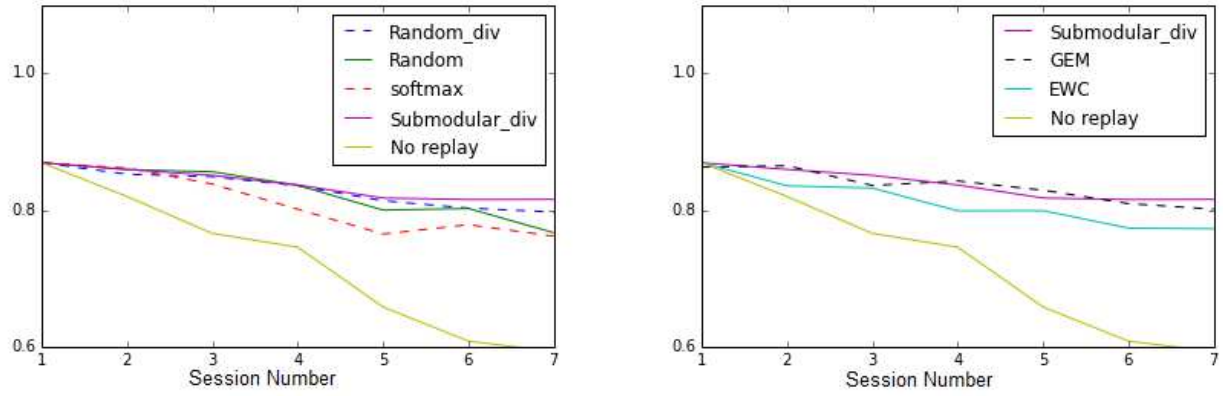
where P_j refers to partitions of the original set V into K separate clusters. It is up to a designer on how to do the partitioning. The diversity reward $D(S)$ is added to $f(S)$ with a weight multiplier to ensure diversity while maximizing coverage.

The diversity constraint can also be put by dividing the set into partitions and maximizing a submodular function on each of it separately. The subsets obtained from each of the partition are then concatenated to form the final subset. Given a total budget of b for the subset memory, we are bound by the constraint $|S \cap P_j| = b/K$. This makes sure that all the K partitions are *equally* represented in the final subset. This also helps in solving for each partition in parallel and easy to implement in a distributed setting. It is a designer's choice on how to come up with partitions of the data. Through experiments, we show how this can help us in choosing a representative yet diverse subset for enabling continual learning on sequentially incoming data.

4. Experiments

4.1. Learning of new Sessions

MNIST permutation [13] is a popular benchmark for showcasing the ability of continual machine learning algorithms. In each session, the image pixels of the handwritten digits are shuffled as per a randomly generated but fixed permutation pattern. The permutation patterns for each session are independent of each other. We start with the original MNIST handwritten digits in the first and keep shuffling over subsequent sessions. In order to have a fair comparison, we closely followed the setting that was laid out in [21] to compare different types of subset replay algorithms along with some standard stand alone continual learning techniques like EWC and GEM. We used a multi-layer neural network with two hidden layers of 100 neurons each. ReLU and softmax are used for hidden and output layers respectively. Each session was trained with 1000 examples belonging to 10 categories. The batch size was fixed to be 10 and all the algorithms were run for 300 number of gradient update iterations during each session. For all the subset replay experiments, the budget b per session was taken to be 100. The size of the memory in GEM was also taken to



(a) (b)

Figure 2. (a) Comparing different subset replay methods, and (b) Comparing submodular replay with other continual learning methods on the MNIST permutation task

be 100 to have a fair comparison. We ran the experiments for seven sessions in total. The accuracy on the original test MNIST handwritten digits, which is also the test data for Session 1, over various sessions is plotted in Figure 2. The model drastically forgets over time on fine-tuning with the new data only. The plot shows the advantage of performing a diversity (based on classes) encouraged submodular (k-medoids) selection over other forms of subset replays. Softmax uncertainty [2] chooses the subset of samples which correspond to maximum entropy values on their softmax output. But such subsets perhaps led to higher over-fitting of the model to boundary located data points. On comparing with state-of-the-art continual learning algorithms, we found out that both GEM and submodular subset replay had very similar performance with the same budget. However, GEM took more time to train as it has an extra gradient projection step that a simple subset replay does not.

4.2. Improving Continual Learning Methods

GEM uses a randomly chosen memory for each session. Also, EWC uses the sample expectation to estimate the Fisher diagonal. We propose that an informative memory management technique like submodular subset can add value to these stand alone continual learning algorithms too. Only two new sessions were chosen for this experiment but more iterations of gradient updates were conducted during each session. For the subset selection, the submodular function corresponding to the k-medoids optimization was selected along with class conditional diversity constraint on the budget. Figure 3 shows that adding a subset replay helps improve the performance of EWC. The plot only shows the results over two new sessions only but the advantage seems to grow in prominence over time.

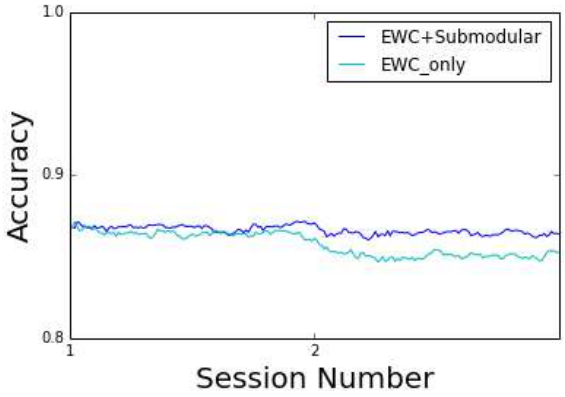


Figure 3. Improving performance of standalone continual learning technique like EWC by adding a submodular subset replay

4.3. Generative replay vs Subset replay

Although the subset memory is expected to grow linearly with respect to the number of sessions, storing actual sensor measurements like images and LIDAR point clouds is still more expensive than just storing a generative model that can learn to spit out likely representations that was seen in each session. This idea has been described in [26] where the authors created generator models, using Wasserstein GAN [10], with the data from each session. For every new session, each of the older generators will be used to fetch data of the older sessions and this will be combined with the training data of the current session to retrain the machine learning model. Figure 4 compares the performance of generative replay as compared to the submodular subset replay. Here, the first session contained all the MNIST handwritten digits corresponding to digits 0 and 1 and the second

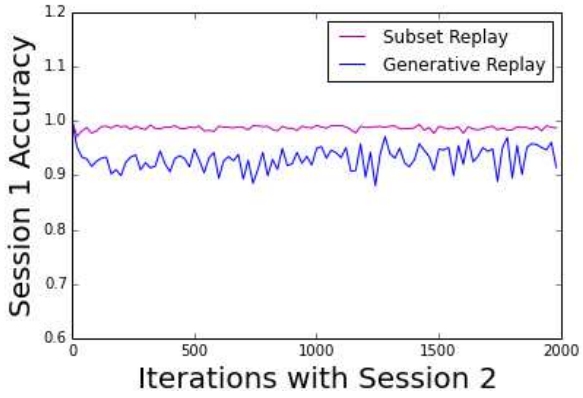


Figure 4. Comparing accuracy on digits belonging to the first session (0 and 1) by using generative replay and subset replay while retraining with newer digits (2 and 3)

session consisted of digits 2 and 3 only. We now trained a WGAN to generate digits belonging to digits 0 and 1 and used it for the generative replay. We selected the budget b in a manner that the overall hard disk memory $\mathcal{M}$ occupied by the subset and the generator model is exactly the same. Subset replay was able to perform better than generative replay as can be seen in Figure 4. On looking at the generated samples, we hypothesize that the GAN was only able to discover limited number of modes or styles in the data. But the diversity encouraged submodular selection was able to pick some of the isolated data points that may not necessarily belong to a mode in the data distribution and thus led to better continuity in learning.

4.4. Learning of Novel Categories

Method	Accuracy on Class 1 to 40
EWC	$92.9\%(-4.5)$
LWF	$86.3\%(-11.1)$
Random Replay	94.4% (-3.0)
Random Balanced Replay	$96.2\%(-1.2)$
Submodular Diverse Replay	$96.9\%(-0.5)$
Joint Training	97.4%

Table 1. Test accuracy on images belonging to the original session after retraining on a new session containing only three new classes of traffic signs. Bracket value indicates deviation from joint training as upper bound.

Traffic sign recognition is a common task in the intelligent systems for a vehicle's perception. The set of traffic signs may change over time and differ from place to place. In the event of collected data containing new categories, a machine learning classification model has to be retrained to cover the increased label space. Continual or lifelong learning can play a major role in aiding this process in a temporally scalable manner. The distribution of the occurrence of traffic signs is often skewed as some signs are more common as compared to others. In order to replicate this scenario, we used the German Traffic Sign Recognition dataset [27]. It has 43 categories of traffic signs training images in total. The dataset has an unbalanced class distribution.

We divided the dataset into two sessions. Session 1 contained all the signs belonging to the first forty traffic signs while Session 2 contained only the last three. This is done to mimic the situation in which novel categories of objects were recorded during a data collection campaign and an existing perception model is then asked to update itself. In Table 1, we compared the performance of various continual learning techniques in remembering the performance on the older categories while learning the novel traffic signs seen in the second session. We used a ConvNet (*Conv0-Conv1-MaxPool0-Conv2-Conv3-MaxPool1-Drop0-Conv4-Conv5-MaxPool2-Drop1-Flat-FC-Drop2-SoftMax*) as the architecture for the classification model. The feature based submodular optimization was done on the features extracted after the last pooling layer. For the subset replay algorithms, the budget b was taken to be 200. Since Session 1 contained 38369 images in total, it is therefore a data reduction to almost 0.5%.

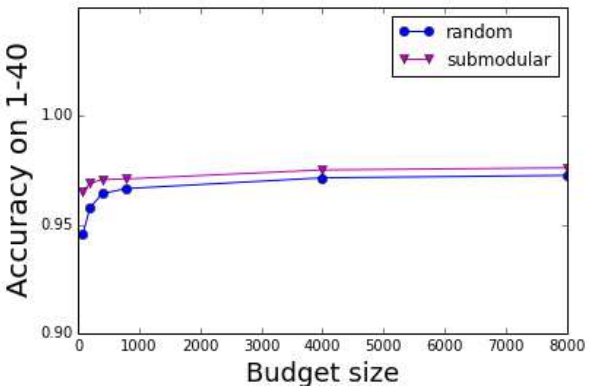


Figure 5. Variation of accuracy by changing the subset budget

On this task, submodular subset selection outclassed all the other techniques that included stand alone continual learning algorithms like EWC and LWF. It is worth pointing out here that the storage requirement for EWC is only for the previous model weights and corresponding Fisher coefficients. For LWF, it is the previous model and corresponding softmax activations on the new input. This is still less as compared to storing a subset of the original dataset. Thus, we took the actual hard disk space $\mathcal{M}$ occupied by these information stored in the h5 format (Keras with Tensorflow backend was used for the experiments) and used

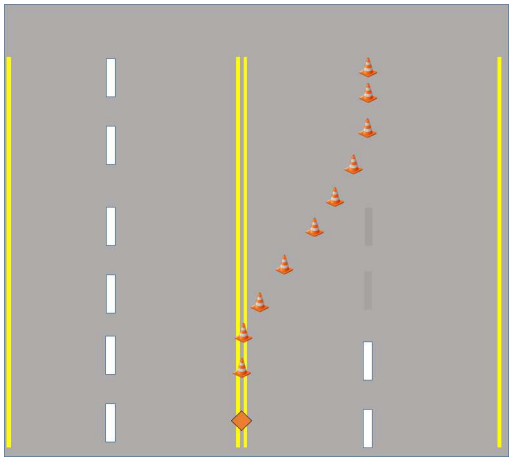


Figure 6. Top view design of the construction zone based corner case for lane keeping. Left lane is closed for construction and the lane markings are partially erased to show merging of two lanes

that to decide the value for the subset memory budget b such that the overall storage consumption is same for all techniques compared in the current experimental setting. Figure 5 shows the performance of the subset replay algorithms by varying the budget hyperparameter. As the budget size increases, the difference in performance between random and submodular decreases and it tends to converge towards the performance obtained by joint training. Thus, submodular selection has the most impact when we are heavily constrained by the budget.

4.5. Learning from Corner Cases

We simulated an anomaly or a corner case situation using the Unity development platform. In this case, the task chosen was automatic lane keeping for which we did imitation learning with a deep convolutional architecture similar to NVIDIA's model[1] to take images as input and produce steering angles for a controller. The regression model was trained end to end from images to steering angles. This was largely inspired by the behavior cloning project in Udacity. The first session depicts a suburban highway with uninterrupted lane markings throughout the entire route. During the training data collection, the car was driven in a lane centered manner at a near constant speed without performing any lane changes. We refer to this as the *normal* session.

A corner case is when the model that was trained on the *normal* case is expected to fail or when a testing driver disengages from autonomous mode to take manual control. A construction zone was designed where cones and traffic signs indicate the closure of left lane. Vehicles on the left lane should merge smoothly into the right lane and those on the right should keep following their lanes. For the construction zone, the broken white lane markings are erased by applying patchy road-colored textures on top. This is described from a top view mode in Figure 6. We now collect data for just that one instance (daytime) where a human driver disengages and manually tries to perform the appropriate merging into the right lane. This session is referred to as the *correction* session as shown in Figure 8. Figure 7 shows the images collected during the training phase of the *normal* session. The *normal* dataset consists of different types of lane curvatures under various types of environments (for example- solar glare, shadows, evening time). The number of recorded images in *correction* session is less than 2% of that of the *normal* session. This is a common situation as anomalous cases are rare and are available in few specific settings only.

To avoid overfitting, we augmented our novel data into various modes. There has been quite a bit of research progress [5] [19] in the area of conditional image generation. Specifically for our experiments, we used CycleGAN [31] which is a two sided GAN based approach to take an image from one domain (sunny day for example) to convert it to another domain (evening or foggy conditions) such that it is indistinguishable to a discriminator trained on the target domain. The data belonging to the *correction* session was collected during daytime only. We trained a CycleGAN on the original *normal* dataset using unpaired images corresponding to each tuple of weather modes. Figure 9 shows how the CycleGAN [31] performed in taking a collected image from the actual daytime *correction* dataset and generated its fake evening time counterpart. We then used the *normal* session with the generated images in addition to the selected subset to train our combined model.

Here, we compare the performance of a lane keeping model with two simple statistical measures, (A) d =Number of undesirable events, and (B) p_{sm} = Probability of occurrence of non-smooth behavior. Undesirable events may refer to disengagements, unnecessary drifting out of the current lane or getting too close to collision objects like cones. On experimenting with a number of test drives, we assign either a high H or low L probability of such disengagements based on a threshold. Assuming the roads to be smooth curves which can be approximated by a piece-wise linear functions, we expect the time series of steering angles in the autonomous mode also to exhibit piece-wise linear behavior. For a short given window of time, a shaky motion or temporally fluctuating steering angles would lead to a higher residual error when we do a linear fitting and will constitute as a non-smooth behavior. Each test drive is divided into small non-overlapping windows of time T each and the residual error is computed on performing line fitting with the steering angles corresponding to that window. If the error is beyond a particular threshold, we consider that as an occurrence of non-smooth behavior. The fraction of such windows over multiple test drives gives us an estimate of the measure p_{sm}. For our

(a) (b) (c)

Figure 7. Front view images belonging to the *normal* session

(a) (b) (c)

Figure 8. Front view images belonging to the *correction* session

Figure 9. Using conditional GANs for data augmentation of one-shot corner case recording into other modes

experiments, we took T to be half a second which corresponds to 15 consecutive steering angles since our sampling frequency is 30. To test the importance of appropriate subset sampling, we compared the measures d and p_{sm} for the situations corresponding to both *normal* $\mathcal{N}$ and *correction* $\mathcal{C}$ sessions. The output range of steering angles was divided into disjoint bins in order to perform feature based submodular selection on each of them separately to promote both coverage and diversity. As can be seen from Table 2, submodular replay does outperform random subset replay in terms of both metrics. Here, 'previous' refers to model that was trained using the *normal* session only.

Method	$p_{sm}^{\mathcal{N}}$	$p_{sm}^{\mathcal{C}}$	$d^{\mathcal{N}}$	$d^{\mathcal{C}}$
Previous	0.14	0.45	L	H
Random	0.27	0.35	H	L
Submodular	0.19	0.34	L	L

Table 2. Statistical measures d and p_{sm} for models trained under different settings.

5. Conclusion

For each acquaintance or event, we tend to remember only a few particularly interesting instances while being able to learn from new ones as we encounter throughout or life. The same intuition was used to come up with a subset replay method to help autonomous systems perform continual learning. We presented a general framework where a user can customize submodular functions and diversity constraints of his or her own choice. We were able to get better results than just using randomly selected subset or other contemporary continual learning algorithms. An intriguing topic that needs further work is to find out what size of subset memory is enough to ensure that there is no degradation of performance while learning over future sessions. This may also play a role in training of GANs to generate more representative samples and can lead to better generative replay. This work can help in scaling up the software improvement life-cycle of autonomous perception modules.

References

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[2] J. Chen, A. Schein, L. Ungar, and M. Palmer. An empirical study of the behavior of active learning for word sense disambiguation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 120–127. Association for Computational Linguistics, 2006.

[3] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[4] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[5] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 2414–2423. IEEE, 2016.

[6] R. Gomes and A. Krause. Budgeted nonparametric learning from data streams. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 391–398. Omnipress, 2010.

[7] B. Gong, W.-L. Chao, K. Grauman, and F. Sha. Diverse sequential subset selection for supervised video summarization. In *Advances in Neural Information Processing Systems*, pages 2069–2077, 2014.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[9] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[10] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779, 2017.

[11] M. Gygli, H. Grabner, and L. Van Gool. Video summarization by learning submodular mixtures of objectives. In *Proceedings CVPR 2015*, pages 3090–3098, 2015.

[12] H. Jung, J. Ju, M. Jung, and J. Kim. Less-forgetting learning in deep neural networks. *arXiv preprint arXiv:1607.00122*, 2016.

[13] 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*, 114(13):3521–3526, 2017.

[14] A. Krause and D. Golovin. Submodular function maximization.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[16] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[17] H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics, 2011.

[18] F. Liu, C. Shen, and G. Lin. Deep convolutional neural fields for depth estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5162–5170, 2015.

[19] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, pages 700–708, 2017.

[20] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[21] D. Lopez-Paz et al. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6470–6479, 2017.

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

[23] F. J. Ordóñez and D. Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115, 2016.

[24] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[25] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[26] H. Shin, J. K. Lee, J. Kim, and J. Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2994–3003, 2017.

[27] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, (0):–, 2012.

[28] P. Stobbe and A. Krause. Efficient minimization of decomposable submodular functions. In *Advances in Neural Information Processing Systems*, pages 2208–2216, 2010.

[29] A. R. Triki, R. Aljundi, M. B. Blaschko, and T. Tuytelaars. Encoder based lifelong learning. *arXiv preprint arXiv:1704.01920*, 2017.

[30] K. Wei, Y. Liu, K. Kirchhoff, C. Bartels, and J. Bilmes. Submodular subset selection for large-scale speech training data. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 3311–3315. IEEE, 2014.

[31] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2223–2232, 2017.