Incremental Online Learning Using Learning Vector Quantization (LVQ)

Mason Smith
1209123041
mosmith3@asu.edu

Chinmayi Shivareddy
1217006544
cshivare@asu.edu

Karan Rahulbhai Shah
1217215649
kshah26@asu.edu

Abhijith Venkatesh Kumar
1217417604
avkumar2@asu.edu

Abdul Rahman Zindani
1210616781
azindani@asu.edu

Abstract

For this project, an architecture for incremental online learning has been implemented to classify images. The architecture is based on work done by Losing et al. [3]. The model utilizes learning vector quantization (LVQ) to generate prototype classifications for observed samples. Placement strategies can use a variety of methods to generate new prototypes, however, for this project, two methods (sampling cost replacement and clustering) have been implemented and compared. Sampling cost replacement relies on an optimization of the cost-function based on a subset of observed samples. Based on the work done by Losing et al. [3], this method was expected to be more accurate than the clustering replacement method.

1 Introduction

Incremental online learning has gained a lot of attention in the recent years. These types of algorithms are highly flexible and can be used for tasks such as image classification and identification even when resources are limited. Incremental online learning has proven to be quite significant due to its adaptive nature and it's many applications in the fields of biomedical data analysis, image recognition, and robotics. These methods are able address the issue of limited memory resources and computational power for the systems that such algorithms may run on. Storing enough prototypes to accurately generate a ubiquitous model might be infeasible for a highly dynamic system. Some systems in image processing may also have long-term trends that would change the usefulness of prototypes (i.e. day/night).

This project focuses on implementing an incremental online learning architecture using Learning Vector Quantization to classify images. Learning Vector Quantization (LVQ) is a family of algorithms for statistical pattern classification, which aims at learning prototypes representing class regions. For these algorithms, training data is used as prototypes in which the object is classified according to the closest resemblance (distance) to the prototype. When a certain number of samples have been misclassified, a new prototype must be introduced to account for these samples. To do this, two different replacement methods, sampling cost replacement and clustering, have been implemented and compared. By replacing the LVQ model prototypes to match relevant object characteristics, the memory needs of the system are reduced while maintaining validity of the finite prototypes.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

2 Feature Extraction

We use the VGG19 [1] Deep Convolutional Neural Network to perform feature extraction from images in our algorithm. The VGG19 network is made up of 19 layers consisting of 16 convolution layers, 3 Fully connected layers, 5 max pool layers and 1 SoftMax layer.

This network takes as input an image of size (224x224x3) which is preprocessed by subtracting the mean RGB value from each pixel of the image. Kernels used in the convolution layers have very small receptive fields of (3x3) with a stride of 1 pixel enabling them to cover the entire image while maintaining the spatial resolution. The network also uses spatial padding to preserve the spatial resolution of the image. The convolution is (1x1) which acts as a linear transformation of the input. Max pooling is performed over a (2x2) pixel window with a stride of 2 to reduce the size of the image as it moves through the network. This is followed by ReLU (Rectified linear unit) to introduce non-linearity and improve the computational ability of the network. There are 3 Fully Connected layers, the first two have 4096 channels each and the third has 1000 channels, 1 for each class. The last layer is a SoftMax layer used to classify images into 1000 classes. The architecture is illustrated in the fig 1 [2] below.

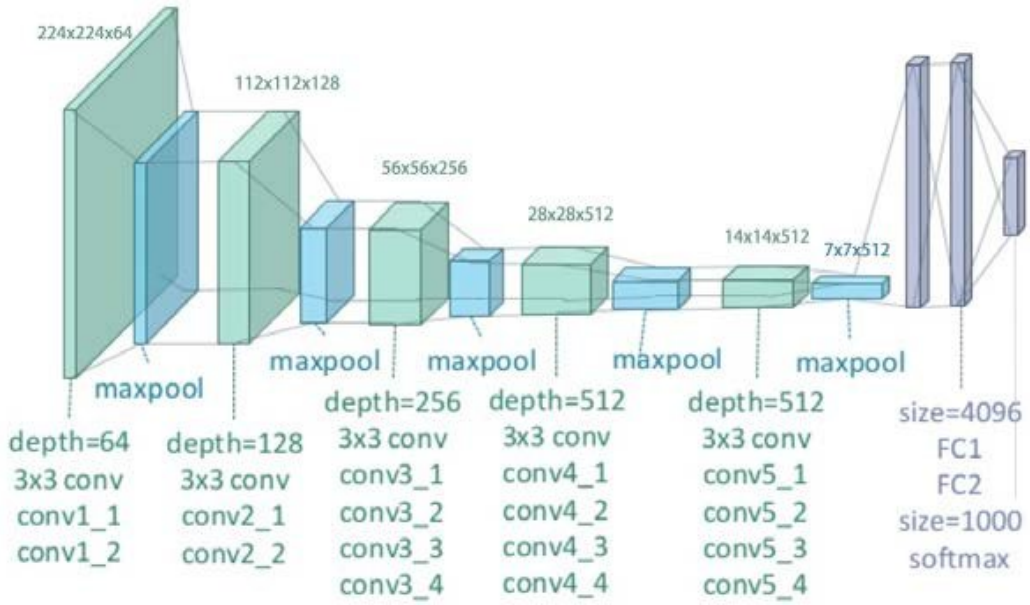

Figure 1: Architecture of VGG19 network.

The VGG19 network is usually used to classify images which is done by the last few layers including the Fully Connected and SoftMax layers, for our application we extract the output from the 5th max pool layer which gives us a matrix of size (7x7x512), containing the condensed features of the image. These features are then utilized to train our online learning model. We do not train the VGG19 network with our dataset as it is a time consuming process, we use the pre-trained weights from the ImageNet dataset which contains the images of common objects.

3 Placement Strategies

For each new sample (x_i, y_i) that the model comes across, the algorithm tests to check if y_i is a new class. If y_i is not already represented within the model, the sample is simply added as a prototype to list of known prototypes. Otherwise, the sample is stored in the short-term memory along with its distance information. If necessary, the new sample may replace old samples due to memory restraints or limitations. If the sample has been classified incorrectly, an error count starts to increment. Whenever a predefined number of errors has been reached, a placement strategy is used to propose a new prototype. This new prototype is then added to the set of prototypes W. Once the

new prototype has been added, the distances within the short-term memory are updated and the error count is reset.

The new prototype may determined based a number of preexisting, state-of-the-art strategies. For one strategy, the misclassified samples are ordered by their distance to the nearest wrong class and this information is used to determine the new prototype. Another strategy involves clustering of misclassified samples. To cluster the incorrectly classified samples, k-Means are used with $k = \sqrt{n/2}$. The centroids of the biggest clusters are selected as new prototypes. A third strategy involves determining the Voronoi region that contains the most misclassified samples and using the mean of the samples within the region to choose the prototype.

When a placement strategy proposes a new prototype (w, l) and it is inserted into the network (i.e. $W := W \cup (\mathbf{w}, l)$), ψ is updated using the following method:

$$\begin{aligned} &\forall(\mathbf{x_i}, y_i, d_i^+, d_i^-) \in \psi : \\ d_i^+ &:= d(\mathbf{x_i}, \mathbf{w}), if y_i = l \wedge d_i^+ > d(\mathbf{x_i}, \mathbf{w}) \\ d_i^- &:= d(\mathbf{x_i}, \mathbf{w}), if y_i \neq l \wedge d_i^- > d(\mathbf{x_i}, \mathbf{w}) \end{aligned} \tag{1}$$

3.1 Sampling Cost

Rather than using the pre-existing, state-of-the-art placement strategies mentioned above,our model uses a strategy called SamplingCost replacement. This strategy utilizes an optimization of the GLVQ-cost-function. This cost function is approximated by the sample ψ.

In order to implement this strategy, a random subset, $\hat{\psi}$, of size $\hat{t}$ is taken from a sample ψ. For every candidate within the subset, the cost is approximated using the following steps:

1. Extend $\hat{W} := W \cup (x_i, y_i)$
2. Update ψ as described in 1, but store the result temporarily in another list ψ'
3. Calculate the cost value $E(\psi', \hat{W})$ on the basis of distances stored in ψ'

The sample with the smallest cost function is then added as a new prototype to W and ψ accordingly. This entire procedure has a time complexity of $O(\hat{t} * t)$. The pseudo code for the Sampling Cost function is shown below.

function SAMPLINGCOST($\psi, \hat{t}$)
minCost := 1
proto := null
for all $(x, y) \in \hat{\psi}$ **do**
  $\psi' := updateShortTermMemory(\psi, (x, y))$
  $cost := calculateCost(\psi')$
  if $cost < minCost$ **then**
    $minCost := cost$
    $proto := (x, y)$
  end if
end for
return proto

3.2 Cluster Replacement

The cluster replacement method that was used in this project follows from the implementation of an adaptive prototype replacement for LVQ [4]. This method focuses on adding prototypes at positions of large clusters of missclassified points and removing prototypes from inappropriate positions in order to optimize the accuracy of the replacement algorithm. To do this, the algorithm scores each prototype to determine if it is likely harmful or beneficial to the resulting accuracy. Consequently, the cluster replacement method uses two functions $LVQremove$ and $LVQadd$ to first remove harmful prototypes then add likely beneficial prototypes to codebook.

First, an initial set of prototypes and labels $\{(p_j, c_j), j = 1, ...p_0\}$ must be defined. These prototypes can be randomly chosen or initialized through another algorithm but must be within a defined prototype budget B. This value sets an upper bound, or prototype to the amount of prototypes that can be included in a codebook. The codebook may contain less prototypes but requires other prototypes to be removed before new ones are added if the number of prototypes is equal to the budget. Then, a new set of training prototypes and labels $\{(m_j, c_j), = 1, ...P\}$ is introduced where the algorithm iterates, for each vector m_j, through scoring prototypes accordingly. During each iteration, prototypes are dynamically added and removed based upon whether the prototype in question will likely result in an increase to the accuracy of the algorithm. More detailed descriptions of this process are contained in the following sections. The algorithm stops either when either the max number of iterations I is reached $it = I$ or there is no significant change in accuracy to the algorithm detected $error_{it} - error_{it-1} < error_{significant}$.

A method to further increase the accuracy in this algorithm involves updating the existing prototypes by moving them away from incorrectly classified points and towards points that they would correctly classify. For an update like this to occur, the class of a prototype closest to training data point x must be different different from the class of the training data point, the class of the next closes prototype must match the class of the training data point, and the training data point must fall near the hyperplane at the midpoint between the closest and the second closest prototype. The update process can be described analytically through the following equations:

$$m_j(t+1) = m_j(t) - \alpha(x - m_j)$$

$$m_k(t+1) = m_j(t) + \alpha(x - m_k)$$

where m_j is the closest and m_k is the second closest prototype to training data point x and α is a monotonically decreasing function of time.

3.2.1 LVQremove

$LVQremove$'s objective is to detect which prototype's removal would likely result in an increase of accuracy of the current codebook. Prototypes like this are typically described by either having little effect on the classification of points (outliers) or located in an position such that its classification area contains a majority of points of a different class. During the process of removal, each prototype m is scored:

$$Score_j = A_j - B_j + C_j$$

where A_j describes how many times m_j classified a point correctly and has not been moved, B_j describes how many times it was moved away as the prototype of the wrong class and C_j describes how many times it was moved towards as the prototype of the correct class. Prototypes with negative scores are then removed from the codebook thereby allowing room for possibly more beneficial prototypes in the budget.

3.2.2 LVQadd

$LVQadd$ aims to find clusters of misclassified points within the same class and add the centroids of these clusters to the codebook until the prototype budget B is reached. The clustering algorithm that was used for this project groups missclassified points by class through heiarchical clustering [5] to locate clusters of size n_j. These clusters are then concatenated into a cumulative list and sorted according to their size. The largest prototypes of size n_j, up until the prototype budget B is reached, are then added to the current codebook. After the prototype budget is reached, this part of the algorithm ceases operation until there is more room for prototypes to be added to the budget.

One of the benefits of using clustering as a replacementment methods is that it allows us to set an upper bound, max clusters, and a lower bound, minimum cluster size, on the clustering method. Tuning these parameters allows us to provide additional specification on the resolution and generality of the algorithm as a whole.

4 Analysis

Analysis of the performance of each replacement method was measured by calculating the accuracy of the independent sampling cost prototypes and cluster replacement prototypes for each iteration using an independent set of test images of the same class. The two tandem algorithms were sequentially provided a set of training images where the user was asked to label it themselves or predict the label using the algorithms and correct it if need be. Table 1 describes the distribution of number of images where the label was given by the user or the label was predicted by the algorithm for each class. These images were then randomly sampled without replacement during each training iteration.

Table 1
Training Image Description

Image	Class of Image	Class Provided	Class Not Provided
Book	0	30	10
Car	1	30	10
Chair	2	30	10
Kite	3	30	10

5 Results

The final accuracy for sampling cost replacement was 53% while cluster replacement achieved an accuracy of 51%. During the initial training iterations it is clear that the sampling cost method has high variation in it's prediction accuracy while the cluster replacement algorithm slowly decays to its stable equilibrium. It should be noted that the computational time for the clustering algorithm is significantly greater and the trend that is seen in Figure 2 would likely not hold for longer number of iterations.

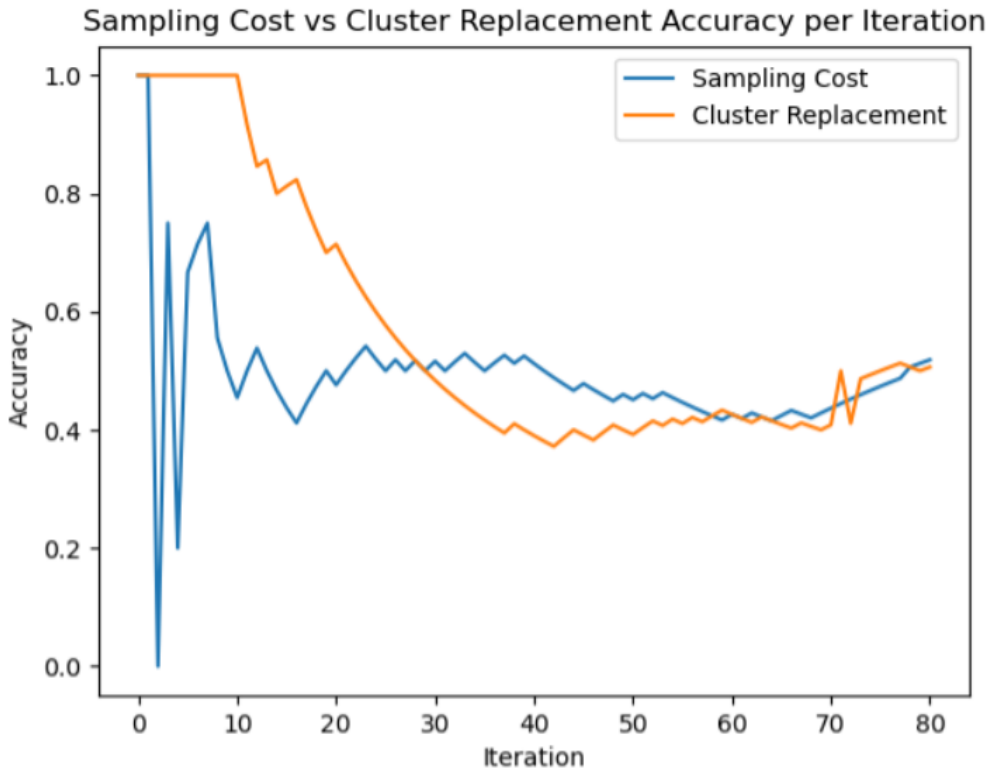

Figure 2: Sampling Cost and Cluster Replacement Accuracy

6 Conclusion

Each algorithm had similar performance in regards to final accuracy and it can be concluded that there would be nominal differences between the two during application into an online learning environment. However, this only holds true over the number of iterations used for this test. Due to the higher computational requirements of cluster replacement, prolonged periods of training may cause deficits in update speed. These tests were also performed on a platform that did not have any applicable limitations on available computational resources and if there were restrictions, cluster replacement would likely struggle. Therefore, the similar performance of sampling cost replacement and lower computational demand would lead to the conclusion that implementing this method would be result in higher performance across all types of robotic platforms.

References

[1] Simonyan, K. Andrew Zisserman. (2015) Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR abs/1409.1556 (2015): n. pag.*

[2] Architecture of VGG19 network: *https://www.researchgate.net/figure/llustration-of-the-network-architecture-of-VGG-19-model-conv-means-convolution-FC-means*$_{f}ig2_{3}25137356$

[3] Losing, Viktor Hammer, Barbara Wersing, Heiko. (2015). Interactive online learning for obstacle classification on a mobile robot. 1-8. 10.1109/IJCNN.2015.7280610.

[4] M. Grbovic and S. Vucetic, "Learning Vector Quantization with Adaptive Prototype Addition and Removal," in Proceedings of International Joint Conference on Neural Networks, Atlanta, 2009.

[5] Seo, B. Shneiderman, Interactively exploring hierarchical clustering results, IEEE Comp uter,pp. 80-86 , Vol. 35,2002.