Abstract—Neural networks suffer from catastrophic forgetting problem when deployed in a continual learning scenario where new batches of data arrive over time; however they are of different distributions from the previous data used for training the neural network. For assessing the performance of a model in a continual learning scenario, two aspects are important - (i) to compute the difference in data distribution between a new and old batch of data and (ii) to understand the retention and learning behavior of deployed neural networks. Current techniques indicate the novelty of a new data batch by comparing its statistical properties with that of the old batch in the input space. However, it is still an open area of research to consider the perspective of a deployed neural network’s ability to generalize on the unseen data samples. In this work, we report a dataset distance measuring technique that indicates the novelty of a new batch of data while considering the deployed neural network’s perspective. We propose the construction of perspective histograms which are a vector representation of the data batches based on the correctness and confidence in the prediction of the deployed model. We have successfully tested the hypothesis empirically on image data coming MNIST Digits, MNIST Fashion, CIFAR10, for its ability to detect data perturbations of type rotation, Gaussian blur, and translation. Upon new data, given a model and its training data, we have proposed and evaluated four new scoring schemes, retention score (R), learning score (L), Oscore and SP-score for studying how much the model can retain its performance on past data, how much it can learn new data, the combined expression for the magnitude of retention and learning and stability-plasticity characteristics respectively. The scoring schemes have been evaluated MNIST Digits and MNIST Fashion data sets on different types of neural network architectures based on the number of parameters, activation functions, and learning loss functions, and an instance of a typical analysis report is presented. Machine learning model maintenance is a reality in production systems in the industry, and we hope our proposed methodology offers a solution to the need of the day in this aspect.

I. INTRODUCTION

Neural networks suffer from catastrophic forgetting[1] problem when trained upon a new batch of data. Catastrophic forgetting is a well-studied problem and many attempts have been made to understand and solve the problem [2][3][4][5][6]. In a Continual learning scenario, neural network is expected to learn new batches of data sequentially without forgetting. Concept Drift [7] is a phenomenon that occurs in continual learning scenarios where, the underlying distribution of the new incoming data changes over time because of the changing dynamics in the real world. When a new batch of data from a different distribution is introduced to a pre-trained neural network, the network tends to forget the previously learnt information.

The reason for this forgetting can be attributed to how the back-propagation algorithm [8] updates the neural network’s weights. When the neural network is trained on a new batch of data, the weights are optimized such that the prediction error for this new distribution is minimal. This greedy approach of optimizing the network weights for the new batch will result in a decision boundary that erroneously classifies data points belonging to previous batches.

If the new batch of data belongs to the same data distribution as the previous batch, it would be similar to re-training the network on the old batch. Hence, during deployment, it is crucial to have tools and techniques that can help the user assess the similarity between the previous and newly arrived data batches. Also, having tools and techniques that provide insights into the neural network’s possible retention and learning behavior when trained upon this new batch of data will be valuable.

A. Previous works

Dataset similarity techniques that provide insights into the similarity between two given datasets can be re-purposed for finding similarities between two given batches of data that arrive over time. For instance, ([9]) quantified the similarity between data sets by representing each data set as a vector embedding. The vector embedding of a data set is obtained by training a CNN on that dataset and then finding the Fisher Information Matrix (FIM) of the CNN’s filters. The process is repeated for the other dataset using an identical CNN. Since identical CNNs are used for both the datasets, the resulting FIMs are of the same size. Statistical techniques could then be employed on the obtained embeddings to find the similarity between given datasets. Overall, it is common practice to compare the similarity between two datasets by first approximating datasets to known probability distributions and then using statistical techniques such as KL Divergence [10] or Bhattacharyya Distance [11] to find their similarity [12]. Even though comparing statistical properties of two data batches like mean, variance or relative information entropy is helpful is comparing the underlying data distributions in the input
dimensional space, it is not sufficient as it ignores the ability of the deployed neural network to generalize on unseen data samples. Two data points may be very proximal in their Euclidean space, however, they may be quite apart from the model point of view. The other way around is quite common as well, where two data points may be apart in their input dimensional space, however, they are similar from the model point of view. Involvement of a model is equivalent to implicitly changing the dimensionality of the data points to either higher dimensions or lower dimensions as dictated by the loss function and training data and carving out a complex decision boundary through the points. The scenario where statistical metrics on input data points do not capture the similarity or dissimilarity between point sets is, when the clusters of points formed based on predicted label correspond to highly convoluted in the given input dimensional space. For example, consider a scenario where a neural network trained on MNIST Digits was deployed. Now a new batch of training data is available where the samples are rotated versions of MNIST digits. If the deployed neural network is a standard Convolutional Neural Network (CNN) [13], then the new batch of data can be considered as novel, as CNNs do not have rotational invariance capabilities. However, if the deployed neural network is a CNN with a data augmentation module which trains the CNN to be rotation invariant by generating rotated samples of the training data during training, then the new batch of data is not novel. If traditional techniques were used directly where the statistical distance between new and old batches of data is obtained by approximating them as some probability distribution, then the deployed neural network’s perspective is completely ignored in the analysis. This would result in the techniques either classifying the new batch of data as either being completely novel or not novel solely based on their statistical properties.

In this work, we propose a dataset similarity measuring technique that considers the deployed neural network’s perspective while calculating the novelty of a new batch of data. Also, we propose a framework called Stability-Plasticity analysis, which is a four-step approach to quantify the possible stability and plasticity nature [14] of the neural network when trained upon the new batch of data. This quantification of the stability-plasticity nature can also allow for a qualitative comparison of various parameters and learning schemes for a continual learning setting [15]. There is initial work in [16] and [17] to provide ways to assess the retention and learning capacities of architectures, however, it is still an open problem to develop a comprehensive framework to judge the novelty of newly arriving data while considering the perspective and potential retention behavior of the deployed neural network.

We demonstrated the proposed dataset similarity technique’s effectiveness by assuming a continual learning setting where the new batches of data were perturbed copies of the initial old data. Two neural networks were deployed in the setup in which one network was capable of rotational invariance while the other was not. The proposed technique was able to identify the degree of novelty of the new incoming data while

Fig. 1: Histograms obtained on normal and rotated versions of MNIST Fashion, CIFAR10 for normal CNN and data augmented CNN. Figures 1a, 1b, 1c, and 1d show perspective histograms for CIFAR10 datasets on normal CNN and rotation invariant CNN. Figures 1e and 1f show perspective histograms for MNIST Fashion for normal and rotation invariant CNN.
considering the two networks’ perspectives. The proposed stability-plasticity scores were demonstrated by analyzing the retention and learning behavior for different continual learning schemes (like Elastic Weight Consolidation [2], Generative Replay [18] using Generative Adversarial Networks [19] and Rehearsal [20]), learning laws (like Backpropagation and STDP) and comparing various neural networks by altering their parameters like network size, batch size, and activation function.

B. Our contributions

In this work, we propose a novel framework for model maintenance in a continual learning scenario having two main components,

- perspective histogram based technique to judge the novelty of the new incoming data while considering the perspective of the deployed model and
- four new scores for R-score, L-score, Oscore, SP-score to understand the stability and plasticity behavior of the deployed neural network for the new batch of data.

II. METHODS

A. Dataset similarity technique

Consider a neural network \( M^1 \), which was trained on \( B_1 \) batch of data. When a new batch of data \( B_2 \) arrives over time, instead of directly training \( M^1 \) on \( B_2 \), \( M^1 \) is made to make predictions on the new batch \( B_2 \). All the correctly classified points and wrongly classified points are segregated into bins based on the confidence with which the model assigned them a class. In this case, we refer to the output of the Softmax layer when we use the word confidence. A confidence interval of magnitude five was taken, which resulted in 18 bins. Nine bins for correctly classified points and nine bins for wrongly classified points. The bins related to correct predictions were named from A to I, and the bins representing the wrong predictions were named from J to R. Next, a histogram is created out of these bins where the x-axis is the bins, and the y-axis is the number of points in the respective bin. The process is repeated for the \( B_1 \) where \( M^1 \) is made to make predictions on samples of \( B_1 \). A similar histogram for the \( B_1 \) is also created. The obtained histograms are then normalized so that the summation of bins is equal to 1. We call these histograms as Perspective Histograms as they express the neural network’s perspective on the batch of data. Examples of the proposed Perspective Histograms before being normalized are shown in Figure 1. Finally, the similarity between the obtained Perspective Histograms can be obtained using standard statistical techniques like KL divergence, Bhattacharya Distance, or Bhattacharya coefficient. The choice of the statistical technique can be influenced by the use case in which the proposed technique is being deployed. For example, to implement an alarm that alerts the arrival of a significantly novel batch of data, the Bhattacharya coefficient could be used as it gives a bounded value between 0 and 1, and a threshold can be easily set, unlike in the case of using Bhattacharya distance or KL Divergence.

The proposed technique is better suitable for continual learning scenarios as it considers the opinion of the deployed neural network \( M^1 \) on the new batch of data \( B_2 \). The rationale for building histograms with confidence bins instead of a direct comparison of the number of wrong and correct predictions is that considering the model’s confidence on each prediction adds an extra dimension of the model’s perspective to the data batch under analysis.

B. Stability-Plasticity (SP) Analysis

In this work, we propose a new framework that quantifies the stability-plasticity nature of a neural network on a given batch of data. The framework is called Stability-Plasticity analysis and has four main metrics namely \( R \), \( L \), \( \phi \), and \( O \), which quantify the retention and learning capacities of a neural network. In the first and second steps, the retention score \( (R) \) and the learning scores \( (L) \) of the neural network are calculated. These values are used in the third step to calculate SP Quotient\( (\phi) \), which exposes the stance of the architecture on stability-plasticity scale. The \( O \) is calculated in the fourth step, which ranks architectures based on their overall retention and learning capacities. This \( O \) can be used to recommend architectures based on their overall performance that are much more suitable to work on the new batch of data.
Computing retention quotient

1) Step 1: Retention score ($R$): A neural network is subjected to an Incremental batch learning experiment, where the network is expected to learn different batches of data sequentially. Consider a batch of data $B_1 = \{(x_i, y_i) | y_i \in D^K\}$, where $D$ is a set of binary vectors of $K$ dimensions where only one of the bits is 1 while remaining all are 0’s. After training the network on the batch $B_1$, it is then sequentially trained on subset of samples from batch $B_2$. Accuracy of the network on $B_1$ is measured after every epoch of training on $B_2$. A graph is plotted with the x-axis as the epoch number and y-axis as the accuracy on the batch $B_1$ and is called as Retention graph. Figure 2a demonstrates the concept of a retention graph. Please refer to Figures 3 and 4 for real examples of retention graphs. The entire procedure is described in Algorithm 1.

Algorithm 1 AlgoRetention($M^0_C, B_1, B_2, E$) //Algorithm for computing retention quotient

Require: Let the initial model trained on batch $B_1$ be, $M^0_C$, and $B_2$ is the new batch of data.

1: //The model is trained on incoming batch $B_2$ and accuracies are computed at each epoch
2: //$E$ is the number of epochs
3: //Let $L = [\]$ be the list of accuracy scores
4: for $i = 1, \ldots, E$ do
5: $M^i = Train(M^{i-1}, B_2)$ //Where $Train(M, X)$ denotes training of a model $M$, on a data set $X$
6: $L = L \odot Acc(M^i, B_1)$ //Where $Acc(M, X)$ denotes accuracy of model $M$ on a data set $X$, concatenated to the list
7: end for
8: return $L$

Retention score ($R$) can be obtained by calculating the ratio between summation of accuracies of the current architecture and that of an Ideal architecture ($M^0_I$). When we say ideal architecture, we mean a neural architecture capable of retaining all the information that it previously learnt without being prone to catastrophic forgetting. When the retention graph is plotted for the ideal architecture, the plot will be a straight line as there will be no significant information degradation. Hence, when we compute the ratio between the summation of accuracy for an ideal architecture with the summation of the current architecture’s accuracy, we can quantify the neural network’s retention capacity. Retention score can be calculated as

$$R = \frac{A_R}{I_R}$$

where,

$$A_R = sum(AlgoRetention(M^0_C, B_1, B_2, E))$$

and is called the Actual Retention and

$$I_R = sum(AlgoRetention(M^0_I, B_1, B_2, E))$$

which is called as the Ideal Retention. $B_1$ and $B_2$ are the training data batches that are trained on the current model $M^0_C$ and Ideal model $M^0_I$ for $E$ number of epochs.

2) Step 2: Learning score ($L$): Similarly, accuracy on a different subset of $B_2$ is calculated at every epoch while being trained on subset of $B_2$ and is plotted on a similar graph called Learning graph. Refer to Figure 2b for concept of a learning graph. The procedure is described in Algorithm 2.

Algorithm 2 AlgoLearning($M^0_C, B_1, B_2, E$) //Algorithm for computing Learning quotient

Require: Let the initial model trained on $B_1$ be, $M^0_C$, and $B_2$ is the incoming batch

1: //The model is trained on incoming batch $B_2$ and accuracies are computed at each epoch
2: //$E$ is the number of epochs
3: //Let $L = [\]$ be the list of accuracy scores
4: for $i = 1, \ldots, E$ do
5: $M^i = Train(M^{i-1}, B_2)$ //Where $Train(M, X)$ denotes training of a model $M$, on a data set $X$
6: $L = L \odot Acc(M^i, B_2)$ //Where $Acc(M, X)$ denotes accuracy of model $M$ on a data set $X$, concatenated to the list
7: //Note that this algorithm only change is in $B_2$ as against (Algorithm 1)
8: end for
9: return $L$

This Learning graph can be used to calculate the learning capabilities of a neural network by again considering the ratio
between the summation of accuracies of an Ideal architecture with current architecture. This ratio is called the Learning capacity \( L \) and is calculated as,

\[
L = \frac{A_L}{I_L}
\]  

(4)

where,

\[
A_L = \text{sum}(\text{AlgolLearning}(M_0^1, B_1, B_2, E))
\]  

(5)

and is called as the Actual learning and

\[
I_L = \text{sum}(\text{AlgolLearning}(M_1^1, B_1, B_2, E))
\]  

(6)

There is no hard and fixed rule on what learning behavior can be considered as ideal learning behavior for a neural network. In general, a neural network is said to have demonstrated a good learning capacity when it reaches the maximum possible accuracy in the least number of epochs. For this text, we considered an ideal behavior where we assumed that the neural network would reach the maximum accuracy at the second epoch itself. However, the proposed metric allows the reader to consider some other learning behavior as ideal. It also has to be noted that, increasing the size of subset \( B_2 \) on which the experiment is being run results in a better assessment of retention and learning capacity of the network.

3) Step 3: Calculation of \( \phi \): After calculating the Learning score and Retention score of the architecture, we can now place the architecture on the stability-plasticity scale. If the architecture demonstrates high retention capacities while showing low learning capacities, then the architecture will rest towards the stable side of the scale. In contrast, if the architecture demonstrates high learning capabilities while demonstrating no retention capacity, the architecture sits towards the plastic side of the scale. To accomplish this, we use the SP Quotient represented using \( \phi \). \( \phi \) is the difference between the retention and learning capacities of the neural network.

It can be calculated using the formula

\[
\phi(R, L) = R - L
\]  

(7)

or

\[
\phi(A_R, I_R, A_L, I_L) = \frac{A_R}{I_R} - \frac{A_L}{I_L}
\]  

(8)

Here, \( R, L \) are the retention score and learning score that we discussed earlier.

The value of \( \phi \) ranges from -1 to 1, where -1 signifies that the learning law is too plastic, and +1 signifies that the learning law is too stable. If \( \phi \) is greater than 0, then the network can be said to be stable as it can retain information but is facing difficulty in learning new information. If \( \phi \) is lesser than zero, then that means that the learning behavior of the network is greater than the retention capacity, and the network is on plastic side of the stability-plasticity scale. An architecture with \( \phi = 0 \) can be considered as being the ideal architecture as it means that the architecture is good at balancing both learning and retention behavior.

4) Step 4: Calculation of \( Oscore \): It has to be noted that, although \( \phi \) can be used to analyze the stability-plasticity nature of a given architecture, it does not provide a complete picture about the network’s capabilities. Consider this example, assume two neural networks \( N_1 \) and \( N_2 \) with retention and learning quotients as \( R_1, L_1 \) and \( R_2, L_2 \) where \( R_1 = 0.2, L_1 = 0.2 \) and \( R_2 = 0.9, L_2 = 0.9 \). In this case, the SP Quotient for both \( N_1 \) and \( N_2 \) is 0. However, that does not mean that both architectures have equal capabilities. \( N_2 \) has better performance as it has demonstrated higher magnitudes of retention and learning scores.

Considering this scenario, an additional metric called \( Oscore \) is being proposed here, which will consider the magnitudes of retention and learning capacities of the network under consideration. \( Oscore \) can be calculated as

\[
Oscore = A_R + A_L
\]  

(9)

\( Oscore \) can also be used to suggest architectures that demonstrate superior retention and learning capabilities. Unlike \( \phi \), \( Oscore \) is not a bounded quantity and can vary depending on the number of epochs chosen during the experimentation process. In general, the higher the \( Oscore \), the better the architecture’s capacity to retain and learn information.

C. Experiments for dataset distance

A total of two experiments have been conducted to validate the proposed technique. The first experiment named Perspecitive capturing experiment is designed to demonstrate that usage of the proposed technique is better suited for continual learning scenario compared to using statistical techniques.
Fig. 5: Figures (5a,5b,5c) are the original versions of MNIST Digits, MNIST Fashion and CIFAR10 presented for as training data to both the CNNs. Figures (5d,5e,5f) are the rotated versions of the datasets that were presented as the test sets.

The second experiment named Data perturbation experiment is designed to demonstrate that the magnitude of difference between the new batch and old batch of data is captured by the proposed technique.

1) Perspective capturing experiment: This experiment was designed to demonstrate the ineffectiveness of using statistical techniques directly in continual learning setting. MNIST handwritten digits, MNIST Fashion and CIFAR10 datasets were used in the experiment. The experiment focuses on the ability to consider the perspective of the machine learning model when finding dissimilarity between datasets. Two CNNs were taken where one CNN had a data augmentation module, which introduced random rotations to the training data resulting in rotational in-variance capabilities while other CNN did not have any such modules. The neural networks were first trained on the normal dataset. A rotated version of the training dataset is now assumed as the new batch of data that arrived over time. Figure (5) shows the training data in the top row and test data in the bottom row. Now, since the first CNN has a data augmentation module, the test data would not provide and specific challenge as the network has rotation invariant capabilities. The same cannot be said for the normal CNN. However, since plain statistical distance techniques do not consider the perspective of the model, they report that the new batch of data is different from the training data or the opposite. The results have been tabulated in Table I. Distances using all three techniques i.e. Bhattacharya coefficient, Bhattacharya distance and KL Divergence have been reported.

2) Data Perturbation Experiment: This experiment was designed to demonstrate the effectiveness of the proposed technique in differentiating different batches of data while considering the perspective of the deployed neural network. In this experiment the original versions of MNIST Handwritten digits, MNIST Fashion and CIFAR10 were used as the first batch of data (B₁). Four different copies of these datasets were created (B₂, B₃, B₄). The first dataset (B₁) was left in pristine condition while the second copy of the dataset (B₂) was added some noise with magnitude one. The third and fourth copies of the dataset (B₃, B₄) were perturbed using the same noise with magnitude two and three respectively. The proposed dataset profiling algorithm was then applied and dissimilarity between first dataset (B₁) with the remaining three datasets (B₂, B₃, B₄) was calculated. It is expected that the distance between datasets increases. The proposed algorithm was tested on different kinds of noise additions including Gaussian blurring, random pixel perturbation and translation of images.

D. Experiments for SP analysis

To demonstrate the proposed scores, we performed two main experiments. In this first experiment, various continual learning models in the literature were realised and SP analysis was performed on them by sequentially training them on MNIST Digits and MNIST Fashion datasets to highlight their stability and plasticity behavior in a continual learning setting. In the second experiment, various parameters of a standard neural network were altered to estimate the stability and
plasticity of the resulting neural network. This experiment is aimed at demonstrating that the proposed metrics can be used for parameter estimation of the neural network as well.

1) Experiment 1: Comparison of CL architectures: This experiment analyzes some prominent continual learning schemes and compares their retention and learning capacities using the proposed metrics. In the Elastic Weight Consolidation (EWC) technique, a conservative weight updation policy is implemented, where the synaptic weights that are responsible for the retention of previous information are protected from drastic updates. This selective approach of updating weights helps the neural network in retaining previous information. In the Generative Replay technique, a GAN is used to generate a synthetic version of previous data, which will be used for rehearsing/retraining the neural network to prevent catastrophic forgetting. The previously mentioned continual learning schemes were compared against two standard Multi-Layer Perceptrons, where one network is allowed to rehearse the previous batch of data, while the other neural network was not allowed to rehearse any previous data. Both the networks used Vanilla SGD [21] for weight updations. The MLP, which does not rehearse on previous data, was considered as our control and was represented as Vanilla SGD in the results.

All the architectures mentioned above were subjected to an Incremental Class learning experiment where they were expected to learn MNIST Handwritten digits and MNIST Fashion datasets sequentially. An average performance over three trials was used to calculate the $R$, $L$, $\phi$, and $Oscore$ values.

2) Experiment 2: Various neural network parameters: In this experiment, various network parameters of the neural networks were altered and their retention and learning behaviors were measured. Network parameters like activation function of the neurons, number of neurons, batch sizes and the learning laws were considered for the experiment. For the activation functions, Relu, Elu, Tanh, Sigmoid, Selu, Exponential, Softsign, Softplus, and Softmax activation functions were implemented in the neural network with 256 neurons at the hidden layer. For analysing the change in retention and learning for different network sizes, the number of neurons was systematically increased from 100 to 2048, in the hidden layer of a dense neural network with Relu activation function. The neural networks were first trained on 60,000 samples of MNIST Handwritten digits and then sequentially trained on 60,000 samples of the MNIST Fashion dataset.

E. Implementation details

All the neural networks realised in the work were implemented in Keras with Tensorflow running in the backend. Google Colaboratory platform was used for running the experiments. The platform used Intel Xeon Dual core CPU with Tesla P100 GPU. All the three datasets, namely MNIST Digits, MNIST Fashion and CIFAR10 were imported from default datasets available in Keras. Both MNIST Digits and MNIST Fashion had 60,000 training samples with 10,000 test samples. Each image had a dimension of 28x28. CIFAR10 on the other hand had 50,000 training samples with 10,000 test samples. Each sample in the dataset was a color image and had a dimension of 32x32x3. For experimental results reported in Table I and Table II, a convolutional neural network with three convolutional layers followed by two dense layers was used. SGD optimizer with categorical cross entropy loss was used to train the neural network. For results reported in Table IV, Table V and Table VI, a standard multi layer perceptron implemented in Keras was used. The Spiking Neural Network was implemented using Bindsnet package [22]. It followed the architectural style reported in [23]. The network had 784 input neurons in the input layer and 100 excitatory neurons in the middle layer. This was followed by 100 leaky integrate and fire inhibitory neurons.

III. RESULTS

A. Perspective experiment

We report here results on data set comparison using perspective histogram method (Table I). As the results suggest, both KL Divergence and Bhattacharya distance only look into the statistical nature of the datasets under comparison. By comparing the statistical properties of the two data batches, KL Divergence indicates that the new batch of data is same as the old batch. In contrast, Bhattacharya distance was able to capture the dissimilarity between the new and old batch of data, but was unable to consider the perspective of CNN with data augmentor which faced no difficulty in recognizing the new batch of data. In the results, the normal CNN was represented using the acronym "CNN" and the CNN with data augmentor is represented with the acronym "R-CNN", where "R" indicates the rotational invariance capability of the network. As row number 3 and 4 suggest, the measured distance with a normal CNN and CNN with data augmentor are different. When we are considering the normal CNN, it clearly indicates that the newly arriving batch is different, but when we consider the CNN with rotational invariance capabilities, the new batches of data are not considered as being any different. When using Bhattacharya coefficient to find the difference between perspective histograms, the value ranges between 0 and 1 where 1 indicates that the two histograms are identical. This property can be exploited to build automatic alarms where a certain threshold could be set to automatically alert the user about arrival of novel data batches.

B. Data perturbation experiment

The results have been tabulated at Table II. The experiment was performed on MNIST Digits, Fashion and CIFAR10 datasets with Gaussian blurring, Random pixel perturbation and image translation operations as synthetic noises. The results clearly show that the dataset-distance between the pristine batch of data $(B_1)$ and other batches $(B_2, B_3$ and $B_3)$ is directly proportional to the magnitude of synthetic noise added to them. The proposed technique was able to capture the difference in the batches of data in all the 9 cases. As KL Divergence is not a bounded score, some values exceed
1, however, the user is free to use Bhattacharya coefficient if a bounded value is needed.

| Dataset     | Noise Added | $B_1$ | $B_2$ | $B_3$ | $B_4$ |
|-------------|-------------|-------|-------|-------|-------|
| MNIST       | Gaussian Blurring | 0.0   | 0.277 | 1.876 | 3.452 |
| MNIST       | Pixel Perturbation | 0.0   | 0.005 | 0.065 | 0.235 |
| MNIST       | Translation     | 0.0   | 0.483 | 3.381 | 4.626 |
| Fashion     | Gaussian Blurring | 0.0   | 0.381 | 1.152 | 1.991 |
| Fashion     | Pixel Perturbation | 0.0   | 0.189 | 0.936 | 1.319 |
| Fashion     | Translation     | 0.0   | 0.329 | 2.147 | 3.391 |
| CIFAR10     | Gaussian Blurring | 0.0   | 0.789 | 0.9106 | 1.022 |
| CIFAR10     | Pixel Perturbation | 0.0   | 0.006 | 0.041 | 0.122 |
| CIFAR10     | Translation     | 0.0   | 0.174 | 0.719 | 1.449 |

TABLE II: Results of the distance experiment using perspective histogram based method.

C. Comparison of CL architectures

The retention and learning capacities offered by various learning capacities were investigated using the proposed four metrics. The highest retentions were observed for Rehearsal technique where the neural network was retrained on the original data while learning the new batch of data. Rehearsing the neural network with the original data can help retain 99.2% of original data and also learn 93.6% of new data.

| Architecture | L   | R   | $\phi$ | Oscore |
|--------------|-----|-----|--------|--------|
| EWC          | 0.83 | 0.813 | -0.018 | 62.162 |
| Vanilla SGD  | 0.859 | 0.503 | -0.355 | 50.892 |
| Generative Replay | 0.874 | 0.983 | 0.109 | 70.431 |
| Rehearsal    | 0.936 | 0.992 | 0.055 | 73.016 |

TABLE III: $R,L,\phi$ and Oscore of various continual learning algorithms. The experiment was performed on MNIST Digits and MNIST Fashion datasets.

D. Comparison of various network parameters

To demonstrate that the proposed metrics can be used for parameter selection, we altered many parameters of a neural network, like the number of neurons in the network, activation functions, batch size, and learning laws. The retention and learning capacities of neural networks for different parameters were computed by subjecting them to incremental class learning experiment where they are required to learn MNIST Digits and MNIST Fashion datasets sequentially. The results have been tabulated in Table (IV, V, VI, VII, VIII). In Table IV, we measured the retention and learning capacities of neural network with varying batch sizes. As the results suggest, the retention capacities of neural networks increase with increasing batch sizes as the number of times the weights of the neural networks are disturbed is minimal. Retention and learning capacities for various activation functions are provided in Table V. Relu activation function, which is currently the most popular choice for neuron activation function, demonstrated highest Oscore. In Table VI, stability and plasticity nature for neural networks with a varying number of neurons in the hidden layer have been calculated. The results suggest that as the number of neurons increases, the representational overlap between new and old information decreases, resulting in increased retention capacities. Table VII compares the retention and learning capacities for neural networks that use standard Backpropagation and STDP learning laws. While the neural network that used STDP learning law demonstrated higher retention capacities, the backpropagation algorithms demonstrated a higher Oscore at the end. In Table VIII, we calculated the retention and learning capacities for neural networks which were subjected to new data batches with varying amounts of overlap. If the overlap between the old and new batch of data is high, then the neural network is expected to forget less. This is visible in the results as the neural network demonstrated the highest retention capacities for scenarios where the overlap between a new and old batch of data was the greatest.

| Batch Size | L    | R    | $\phi$ | Oscore |
|------------|------|------|--------|--------|
| 32         | 0.972 | 0.453 | -0.518 | 13.24 |
| 64         | 0.987 | 0.616 | -0.371 | 14.996 |
| 100        | 0.992 | 0.728 | -0.264 | 16.12 |
| 256        | 0.993 | 0.862 | -0.131 | 17.47 |
| 512        | 0.996 | 0.921 | -0.075 | 18.076 |
| 1024       | 1.000 | 0.952 | -0.04  | 18.49 |
| 2048       | 0.997 | 0.974 | -0.022 | 18.626 |
| 6000       | 1.000 | 0.984 | -0.016 | 18.752 |
| 10000      | 0.999 | 0.987 | -0.012 | 18.675 |
| 60000      | 0.986 | 0.989 | 0.003  | 18.681 |

TABLE IV: $R,L,\phi$ and Oscore for varying batch sizes on MNIST Digits and MNIST Fashion datasets.

| Activation Function | L | R | $\phi$ | Oscore |
|---------------------|---|---|--------|--------|
| Relu                | 0.994 | 0.931 | -0.062 | 18.178 |
| Elu                 | 0.965 | 0.855 | -0.64  | 12.203 |
| Sigmoid             | 0.938 | 0.904 | -0.333 | 8.4866 |
| Tanh                | 0.962 | 0.151 | -0.810 | 10.163 |
| Selu                | 0.959 | 0.195 | -0.764 | 10.573 |
| Exponential         | 0.758 | 0.030 | -0.728 | 7.127 |
| Softsign            | 0.959 | 0.110 | -0.849 | 9.722 |
| Softplus            | 0.956 | 0.030 | -0.925 | 8.914 |
| Softmax             | 0.1  | 0.010 | -0.090 | 1.001 |

TABLE V: The retention and learning capacities of various activation functions for MNIST Digits and MNIST Fashion datasets.
TABLE VI: $R$, $L$, $\phi$ and $O$-score values for different sizes of hidden layer in the neural networks.

| Neurons | $L$ | $R$ | $\phi$ | $O$-score |
|---------|-----|-----|--------|-----------|
| 100     | 0.966 | 0.059 | -0.915 | 9.204 |
| 256     | 0.974 | 0.031 | -0.922 | 9.276 |
| 512     | 0.970 | 0.096 | -0.874 | 9.688 |
| 1024    | 0.968 | 0.103 | -0.865 | 9.739 |
| 2048    | 0.993 | 0.446 | -0.546 | 13.359 |

TABLE VII: $R$, $L$, $\phi$ and $O$-score for Backpropagation and STDP learning laws. The experiment was also carried out on MNIST Digits and MNIST Fashion datasets.

| Learning Law | $L$ | $R$ | $\phi$ | $O$-score |
|--------------|-----|-----|--------|-----------|
| DNN+BP       | 0.868 | 0.208 | -0.66 | 5.828 |
| SNN+STDP     | 0.825 | 0.58  | -0.24 | 3.03 |

TABLE VIII: $L$, $R$, $\phi$ and $O$-score metrics for different overlapping distributions. The first column indicates the number of samples that were common in the new and old data batches. As the magnitude of overlap increases, the retention increases because of rehearsal concept.

| No. of old samples | $L$ | $R$ | $\phi$ | $O$-score |
|--------------------|-----|-----|--------|-----------|
| 10                 | 0.977 | 0.677 | -0.299 | 15.502 |
| 100                | 0.968 | 0.612 | -0.335 | 14.97 |
| 1000               | 0.964 | 0.924 | -0.039 | 17.840 |
| 10000              | 0.974 | 0.980 | 0.006  | 18.47 |

IV. CONCLUSIONS AND FUTURE DIRECTIONS

In this work, we introduced a way to estimate the novelty of a newly arrived batch of data while considering the perspective of the deployed neural network. While currently, practices compare the previous and new data batches by comparing their statistical properties, we present perspective histogram based approach where confidence scores of a deployed neural network and hence it’s perspective, are captured over a given data set. The proposed perspective histograms are a vector representation of the data batches based on the correctness and confidence in the prediction of the deployed model. We have successfully tested the hypothesis empirically on image data coming MNIST Digits, MNIST Fashion, CIFAR10, for its ability to detect data perturbations of type rotation, Gaussian blur, and translation. We also introduced four scoring schemes for assessing the performance of a neural network deployed in a continual learning scenario. The scoring schemes are retention score ($R$), learning score ($L$), O-score and SP score for studying how much the model can retain its performance on past data, how much it can learn new data, a combined expression for the magnitude of retention and learning and stability-plasticity characteristics respectively. We demonstrated the proposed scores on different types of neural network architectures based on the number of parameters, activation functions, and learning loss functions. The scoring schemes have been evaluated over MNIST Digits and MNIST Fashion data sets. Machine learning model maintenance is a reality in production systems in industry, and we hope our proposed methodology offers a solution to the need of the day in this aspect.

REFERENCES

[1] R. M. French, “Catastrophic forgetting in connectionist networks,” Trends in cognitive sciences, vol. 3, no. 4, pp. 128–135, 1999.
[2] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, 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, vol. 114, no. 13, pp. 3521–3526, 2017.
[3] R. Kemker and C. Kanan, “Fearnet: Brain-inspired model for incremental learning,” arXiv preprint arXiv:1711.10563, 2017.
[4] R. Coop, A. Mishal, and I. Arel, “Ensemble learning in fixed expansion layer networks for mitigating catastrophic forgetting,” IEEE transactions on neural networks and learning systems, vol. 24, no. 10, pp. 1623–1634, 2013.
[5] G.-M. Park, S.-M. Yoo, and J.-H. Kim, “Convolutional neural network with developmental memory for continual learning,” IEEE Transactions on Neural Networks and Learning Systems, 2020.
[6] A. Gepperth and C. Karakuzu, “A bio-inspired incremental learning architecture for applied perceptual problems,” Cognitive Computation, vol. 8, no. 5, pp. 924–934, 2016.
[7] S. Wang, L. L. Minku, and X. Yao, “A systematic study of online class imbalance learning with concept drift,” IEEE transactions on neural networks and learning systems, vol. 29, no. 10, pp. 4802–4821, 2018.
[8] H. J. Kelley, “Gradient theory of optimal flight paths,” AJS Journal, vol. 30, no. 10, pp. 947–954, 1960.
[9] A. Achille, M. Lam, R. Tewari, A. Ravichandran, S. Maji, C. C. Fowlkes, S. Soatto, and P. Perona, “Task2vec: Task embedding for meta-learning,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 6430–6439.
[10] S. Kulhback and R. A. Leibler, “On information and sufficiency,” The annals of mathematical statistics, vol. 22, no. 1, pp. 79–86, 1951.
[11] A. Bhattacharyya, “On a measure of divergence between two statistical populations defined by their probability distributions,” Bull. Calcutta Math. Soc., vol. 35, pp. 99–109, 1943.
[12] D. Alvarez-Melis and N. Fusi, “Geometric dataset distances via optimal transport,” arXiv preprint arXiv:2002.09271, 2020.
[13] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” Neural computation, vol. 1, no. 4, pp. 541–551, 1989.
[14] W. C. Abraham and A. Robins, “Memory retention—the synaptic stability versus plasticity dilemma,” Trends in neurosciences, vol. 28, no. 2, pp. 73–78, 2005.
[15] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, “Continual lifelong learning with neural networks: A review,” Neural Networks, vol. 113, pp. 54–71, 2019.
[16] R. Kemker, M. McClure, A. Abitino, T. L. Hayes, and C. Kanan, “Measuring catastrophic forgetting in neural networks,” in Thirty-second AAAI conference on artificial intelligence, 2018.
[17] N. Díaz-Rodríguez, V. Lomonaco, D. Filliat, and D. Maltoni, “Don’t forget, there is more than forgetting: new metrics for continual learning,” arXiv preprint arXiv:1810.13166, 2018.
[18] H. Shin, J. K. Lee, J. Kim, and J. Kim, “Continual learning with deep generative replay,” in Advances in Neural Information Processing Systems, 2017, pp. 2990–2999.
[19] 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, 2014, pp. 2672–2680.
[20] A. Robins, “Catastrophic forgetting, rehearsal and pseudorehearsal,” Connection Science, vol. 7, no. 2, pp. 123–146, 1995.
[21] H. Robbins and S. Monro, “A stochastic approximation method,” The annals of mathematical statistics, pp. 400–407, 1951.
[22] H. Hazan, D. J. Saunders, H. Khan, D. Patel, D. T. Sanghavi, H. T. Siegelmann, and R. Kozma, “Bindsnet: a machine learning-oriented spiking neural networks library in python,” Frontiers in neuroinformatics, vol. 12, p. 89, 2018.
[23] P. U. Diehl and M. Cook, “Unsupervised learning of digit recognition using spike-timing-dependent plasticity,” Frontiers in computational neuroscience, vol. 9, p. 99, 2015.