MULTIVARIATE GAUSSIAN REPRESENTATION OF PREVIOUS TASKS FOR CONTINUAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual learning has recently become increasingly important with the development of deep learning technology. The memory-based rehearsal is one of the dominant methods: It samples data from a previous task, stores them in memory, and retrains them with the current task. However, since the whole data cannot be stored in fixed memory capacity, there is a problem to lose the knowledge of previous data. In this paper, we propose a method for storing and reproducing distributed representations of data for each class in memory. Data representation is categorized by class and converted into a multivariate Gaussian distribution, which is stored in memory in the form of means and variances. A generative algorithm regenerates the model of previous tasks to restore the data representation for the current task. In the inference process, local adaptation adjusts the model to the distributed representation of data that changes as the number of tasks increases. Experiments with CIFAR-10, CIFAR-100, and tiny-ImageNet show performance improvements of 2.2%p, 5.01%p, and 3.44%p, respectively, compared to the state-of-the-art method of memory replay, confirming the effectiveness of the proposed method in data representation for memory replay.

1 INTRODUCTION

Continuous learning is a learning scenario that a model that learns data streams without forgetting previously learned knowledge (Bang et al., 2022)(Bang et al., 2021) (Lopez-Paz & Ranzato, 2017) (Prabhu et al., 2020). Recently, many deep learning models have shown excellent performance in multiple domains, but unfortunately, when the data distribution changes sequentially, the phenomenon of losing previously acquired knowledge occurs, called catastrophic forgetting (CF) (McCloskey & Cohen, 1989). This phenomenon can be alleviated by combining newly added and previous data, but it is inevitable that retraining is inefficient if the model size is too large, and sometimes the model cannot access previous data (e.g.,due to for privacy issues).

As an approach to solving CF, inspired by the principle of human memory, a rehearsal method using memory with a separate space from the model has been investigated (Bang et al., 2022) (Bang et al., 2021) (Prabhu et al., 2020) (Kim et al., 2021) (Kirkpatrick et al., 2017).The method prevents CF effectively by explicitly storing raw data in memory so that the model can preserve knowledge and combine them with data stored in memory when learning other data or tasks. However, traditional rehearsal methods do not capture the variance of each class by storing only some data due to the limitations of fixed memory space. In addition, it is vulnerable to overfitting because it focuses on data only inside the memory. And the method that relies on the data sampling method has the possibility of losing information of the previous tasks when updating the memory, making it challenging to choose which data samples should be discarded (Zhang et al., 2021).

Let $d^k_{discard}$ be a data sample of class k that is not stored in the memory. C and N are a fixed memory capacity, the number of data N respectively, where $N > C$. The distribution of the data set $\sigma(D^k)$ and the distribution of $\sigma(D^{'k})$, $\{D^{'k}|d^k_{discard} \notin D^{'k}, \forall d^k_{sampled} \in D^{'k}\}$ are not the same. Continual learning (CL), when viewed from a long-term perspective, continues to increase the number of data and classes while having a fixed memory capacity, causing the problem of not accurately capturing data variance. Figure 1 shows this problem with the chaning dimension boundaries according to the discard of data samples.

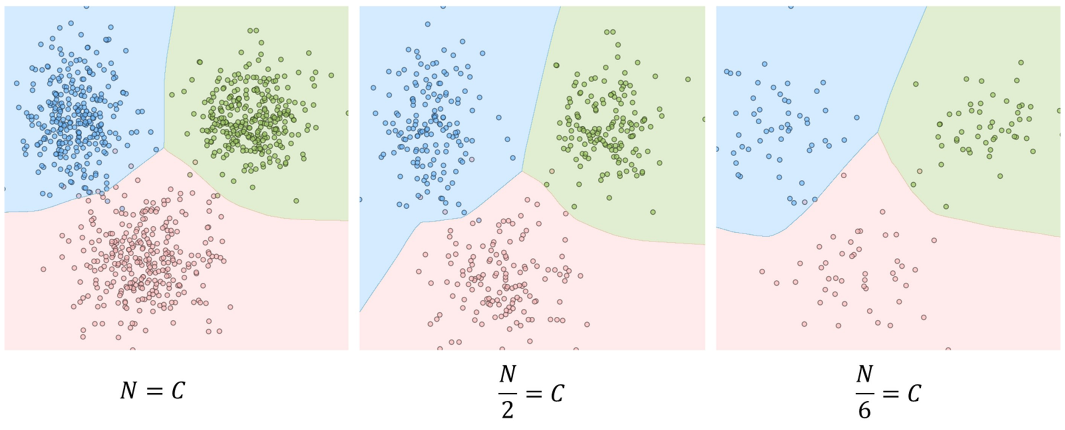


Figure 1: . The dimension boundaries change when random data are discarded and stored in memory; each circle represents raw data. As the difference between N and C enlarges, there is a difference in the original data distribution

In this paper, we propose a method of storing data representation in memory and solving CF using a generative algorithm. The goal of CL is to retain the knowledge that the model has learned in a continuous task until the last task, implicitly maintaining the distribution of all the learned data from the model's perspective. Thus, the key idea of the proposed method is to enable the model to learn and maintain the distribution of data. Based on the idea of Memory Recall (Zhang et al., 2021), we learn the data representation with multivariate Gaussian distribution and construct the model. For learning the distribution, we capture the representation of data, not raw data themselves, as a multivariate Gaussian mixture model, store them in memory for each class, reproduce the data using the generative algorithm in the new task, and combine them with the data in the task to allow the model to learn the distribution.

When storing data representations it is required that the model be consistent in its data representations as it learns the tasks. To satisfy this, we construct the representation model that limits the gradient to learn in a direction orthogonal to the previous tasks for the representation of the data maintain in the same space for each task (Zeng et al., 2019). Additionally, the classification performance decreases where the ranges of each data overlap, resulting in a drop in predictive performance as the model pursues general performance for all data. To solve this problem, we apply local adaptation to prevent it(de Masson D'Autume et al., 2019) (Sprechmann et al., 2018).

We conduct several experiments by changing the number of tasks for the datasets of CIFAR 10, CIFAR100 and tiny-ImageNet, and achieve the best performance compared to the previous models. Additionally, significant performance improvements are derived through memory access and associated representation regeneration in the inference phase.

Our main contributions are as follows (1) Instead of raw data, we store the representation in memory for each class of data and solve the CF by regenerating the data. (2) To solve the uncertainty of the data represented by the distributed expression, local adaptation is used to prevent this. (3) It achieves the-state-of-the-art performance in the datasets such as CIFAR-10, CIFAR-100 and tiny-ImageNet.

2 Related Works

2.1 Rehearsal-based continual learning

Prabhu et al. (2020) suggested a method that can operate in the same way as in the class and task incremental situations. This method, which greedily samples data, stores it in memory, and learns the entire memory as a new model, has inspired many rehearsal methods. Despite its simplicity, the rehearsal-based approach has shown outstanding performance in continual learning settings. Inspired by sampling methods, many studies have been conducted to screen relevant data in renewable memory (Bang et al., 2022) (Bang et al., 2021) (Chaudhry et al., 2018b) (Castro et al., 2018)

(Chaudhry et al., 2018a). Bang et al. (2021) is a method for sampling by estimating the uncertainty of the data , and Yoon et al. (2021) proposed diversified sampling by "Core-Set (Sener & Savarese, 2017)". However, storing raw data samples in memory is not accurate in capturing the distributed representation of a particular class, and excavating necessary data is a task that requires additional effort. In this work, we try to solve the problem arising from raw data sampling by using the distributed representation of each data class. Since our method estimates the variance using all data, we do not need to sample the data. We only allocate additional space to the memory according to the class so that we can expect a minimal memory update.

2.2 Representation learning in continual learning

The continual learning method focusing on representation is as follows. Rebuffi et al. (2017) prevents representations from being forgotten by leveraging distillation and designed for the blurry setup. Javed & White (2019) gives the model a new functionality, OML, to learn a robust learning representation that prevents CF. Gupta et al. (2020) proposes La- MAML (Look-ahead MAML), a fast optimization-based meta-learning algorithm with the help of small episode memory. Both the previous two works attempt to solve CL using data representation in relation to meta-learning.Gallardo et al. (2021) Zhang et al. (2020) applies the self-supervised learning method to generate a more generalized data representation.

2.3 Regularization in continual learning

This method can be divided into limiting the learning of parameters and adjusting gradients. Kirkpatrick et al. (2017) first introduced the Regularization method that limits the changes in the essential parameters of the model, and similar work was proposed in Hu et al. (2021). The gradient optimization approach induces the weight gradient to be orthogonal to input data to prevent changes in the output vectors (Zeng et al., 2019) (Wang et al., 2021) (Saha et al., 2021). Zeng et al. (2019) proposed OWM, which estimates the orthogonal projector to the null space of input. Our main idea is to store the distributed data representation of the task in memory. At this time, the model that outputs the data representation is also affected by CF, so we use OWM to ensure that the model's output is represented consistently.

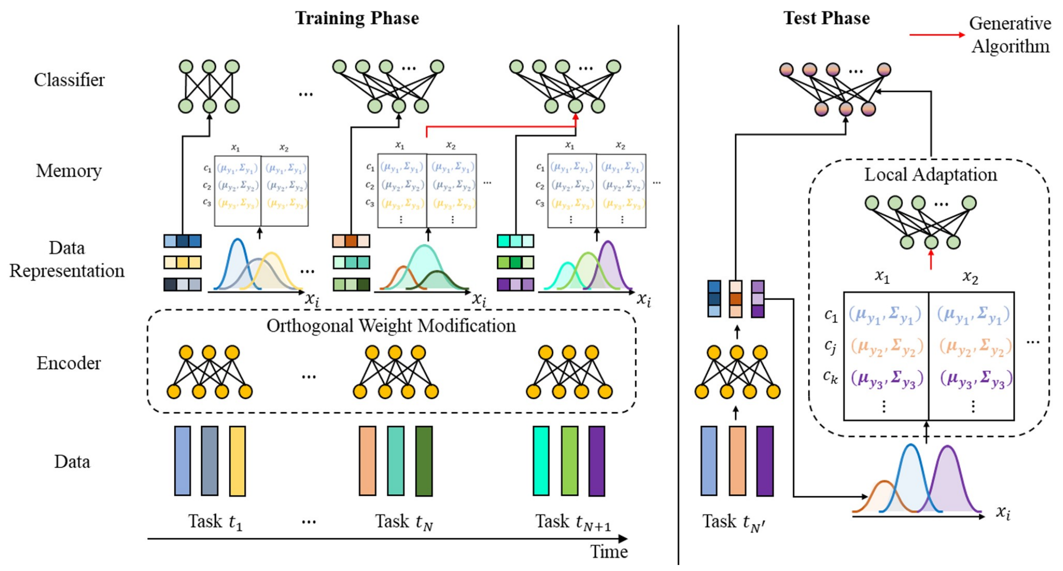

Training Phase
Test Phase
Generative Algorithm
Classifier
Memory
Data Representation
Encoder
Data
Orthogonal Weight Modification
Local Adaptation
x_1
x_2
c_1 $(\mu_{y_1}, \Sigma_{y_1})$ $(\mu_{y_1}, \Sigma_{y_1})$
c_j $(\mu_{y_2}, \Sigma_{y_2})$ $(\mu_{y_2}, \Sigma_{y_2})$
c_k $(\mu_{y_3}, \Sigma_{y_3})$ $(\mu_{y_3}, \Sigma_{y_3})$
x_i
Task t_1
Task t_N
Task t_{N+1}
Task $t_{N'}$
Time

Figure 2: The overview of proposed method

3 PROPOSED METHOD

When task t_n is input, the encoder outputs a data representation. The representation is forwarded as a classifier, and the label is predicted. At the same time, the representation is transformed into a multivariate Gaussian distribution by expectation-maximization(EM) algorithm. In the new task, t_{n+1} state, the means and variances from $t_{0,\dots,n}$ stored in memory are regenerated through the generative algorithm and forwarded to the classifier with the data representation of t_{n+1}. The OWM learning mechanism is used so that the output representation of the encoder can be in a certain space according to the task. In the test step, local adaptation is performed to select a representation similar to the test data from memory and retrain the classifier to eliminate the uncertainty of the distributed representation. The adjusted classifier is then discarded after predicting the test dataset(detailed in 3.4).

3.1 MODEL ARCHITECTURE AND DEFINITION

The network of this method splits into two parts: encoder and classifier. The encoder is a part that learns the general representation of input data, which is constructed as several convolutional layers. The classifier is a part that predicts the input representations into their classes, which is constructed as several Fully connected layers. Denote encoder and classifier as E_i and C_i, respectively, where i represents each model after index learning of the task T_i. Task T_i consists of data $\boldsymbol{x}_i^k$ and a label $\boldsymbol{y}_i^k$, where k represents the class.

3.2 DATA REPRESENTATION WITH GRADIENT REGULARIZATION

We first train the encoder to output meaningful representations from the data. Since the previous task does not exist at $i = 0$, memory contributes nothing to the model's learning. Thus, the model is trained with $\boldsymbol{y}_0^k = C_0(E_0(\boldsymbol{x}_0^k))$. After that, the gradient regularization method is applied, when $i > 0$, to ensure that the output of E_i and the output of E_{i+1} are consistent. The idea is that the gradient of E_{i+1}, ∇W_{i+1} computed from a backpropagation, if $\nabla W_{i+1}^T \boldsymbol{X}_i = 0$, then the output will remain the same, which means the learned knowledge is preserved. So, we must compute an orthogonal projector matrix $\boldsymbol{P}$ that projects ∇W to the orthogonal subspace of input space. By Least Square Solution, It is computed by $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A} - \alpha\boldsymbol{I})^{-1}\boldsymbol{A}$, where $\boldsymbol{A}$ is the set of all output $\boldsymbol{y}_l$ by layer l in E, $\boldsymbol{I}$ is Identity Matrix, and α is a hyperparameter. Because the network cannot access every input vector of each layer in continual learning, OWM utilizes the Recursive Least Square (RLS) Algorithm to approximately compute P by the following equation.

$$\boldsymbol{P}_l = \boldsymbol{P}_{l-1} - \boldsymbol{k}_l \boldsymbol{y}_l^T \boldsymbol{P}_{l-1} \tag{1}$$

$$\boldsymbol{k}_l = \frac{\boldsymbol{P}_{l-1}\boldsymbol{y}_l}{\alpha + \boldsymbol{y}_l^T \boldsymbol{P}_{l-1}\boldsymbol{y}_l} \tag{2}$$

The orthogonal projector $\boldsymbol{P}_l$ of the l-th layer of the network is then multiplied by the weight gradient of the corresponding layer to regularize the learning direction. Our goal is to approximate the data distribution, so we normalize the output vectors of the encoder to reduce the variance of their distribution to help the estimation. Moreover, we add a self-supervised loss to let the network learn generalized features(Shen et al., 2021).

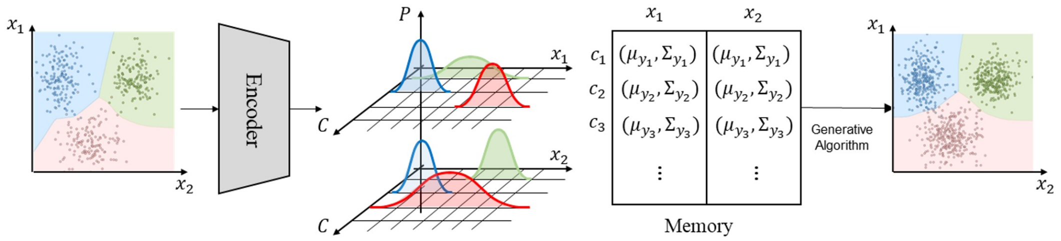


Figure 3: A schematic diagram of the distributed representation and memory storage, regeneration of the data

3.3 Memory Access and Data Representation Generation

After training the network with OWM, we estimate the distribution of representations as a mixture of multivariate Gaussian distribution. Multivariate Gaussian represents data with mean μ and covariance Σ as

$$N(x) = (\frac{1}{2\pi})^{-\frac{p}{2}}|\Sigma|^{-\frac{1}{2}}exp(-\frac{1}{2}(x'-\mu)^T\Sigma^{-1}(x'-\mu)) \tag{3}$$

Where x' is $E(x)$. The advantage of multivariate Gaussian distribution is that the representation can be seen from multiple dimensional perspectives. Covariance shows the correlation between two variables so that multivariate Gaussian can represent the actual distribution accurately. To estimate the such distribution, we use Expectation-Maximization (EM) algorithm to compute means and covariances (Dempster et al., 1977). EM algorithm is a method to find the maximum log-likelihood of probability function as

$$\mathcal{L}(X;\theta) = \ln p(X|\pi,\mu,\Sigma) = \Sigma_x \ln \Sigma_{k=1}^{K}\pi_k N(x|\mu_x,\Sigma_k) \tag{4}$$

Where N is a number of x, K is the number of distributions, $N(\cdot)$ is the equation (3) to compute means and covariances, and π is the ratio of data in each distribution. The EM iterates between two steps. The expectation step (E-step) evaluates the responsibility γ, which is the probability of the data point x_n in the k-th distribution, uses current means and covariances.

$$\gamma(z_{nk}) = \frac{\pi_k N(x_n|\mu_k,\Sigma_k)}{\Sigma_{j=1}^{K}\pi_j N(x_n|\mu_j,\Sigma_j)} \tag{5}$$

γ is then used to compute means and covariances in the Maximization step,

$$\mu_k = \frac{\Sigma_{n=1}^{N}\gamma(z_{nk})x_n}{\Sigma_{n=1}^{N}\gamma(z_{nk})}, \Sigma_k = \frac{\Sigma_{n=1}^{N}\gamma(z_{nk})(x_n-\mu_n)(x_n-\mu_n)^T}{\Sigma_{n=1}^{N}\gamma(z_{nk})}, \pi_k = \frac{1}{N}\Sigma_{n=1}^{N}\gamma(z_{nk}) \tag{6}$$

These equations are derived by partial derivative equation (4) to maximize the likelihood. The computed means and covariances per class are stored in the memory. In the proposed method, the data representation stored in memory has one row per class, and if the number of classes at this time is C, K represents the number of classes, so the size of Memory M becomes $C \times K$.

The covariance and mean stored in memory are repeatedly used to generate data presentation. From randomly generated normal vectors S, we generate the representation $\hat{x}$ of class k as

$$\hat{x}^k = (S * \lambda_k)\Gamma_k^T + \mu_k \tag{7}$$

$$\Sigma_k = \Gamma_k \lambda_k \Gamma_k^T \tag{8}$$

where the covariance is decomposed into an eigenvalue matrix λ_k and eigenvector matrix Γ_k.

Classifier C_i performs classification through the data representation regenerated from the memory and the output of the encoder. To cope with the increase in class, a new classifier C_i is generated for each task to proceed with learning. So,

$$y_{0,1,\ldots,i} = C_i(E_i(X_i); \hat{X}_{0,1,\ldots,i-1}) \tag{9}$$

where $\hat{X}_{0,1,\ldots,i-1}$ is set of $\hat{x}$ from memory. That is since the classifier can learn the representation of data for $T_0, T_1, \ldots, T_i$, CF can be prevented.

3.4 Local Adaptation in Test Phase

When representation is stored and regenerated, uncertainty exists where the distribution between the data overlaps. C and E are linked and affected by loss, and there is a tendency to learn and generalize all data.

We apply a method of fine-tuning the associated with the test dataset, inspired by local adaptation (de Masson D'Autume et al., 2019). Unlike the base method, We use $E(x_{N'})$ as the key value by storing the multivariate Gaussian distribution in memory and selecting the covariance and average of the classes to be regenerated accordingly, where N' is the test task. Within the memory consisting of the mean and covariance, the class with the highest probability of belonging is selected through

Table 1: Results of our method and other baseline methods on CelebA, when trained after different epochs and with different margins.

Method		CIFAR-10 Accuracy(%)	CIFAR-100 5 tasks	CIFAR-100 10 tasks	tiny-ImageNet Accuracy (%)
Joint Learning		73.01±0.74	39.97±0.88		37.15±0.20
Replay	Rebuffi et al. (2017)	58.93±0.19	25.43±0.70	23.61±0.51	26.29±0.47
	Wu et al. (2019)	57.08±0.63	26.84±0.87	14.95±0.37	-
	Chaudhry et al. (2021)	41.57±0.25	29.34±0.49	26.88±0.59	-
	Wu et al. (2018)	22.39±0.83	20.52±0.46	15.23±0.62	13.48±0.16
	Aljundi et al. (2019)	50.95±0.36	24.87±0.72	23.63±0.52	21.84±0.34
Regularization	Kirkpatrick et al. (2017)	18.98±0.10	12.53±0.69	7.56±0.25	7.33±0.30
	Wang et al. (2021)	18.65±0.85	8.69±1.09	6.68±0.88	6.02±0.74
	Zeng et al. (2019)	52.11±0.83	29.93±0.83	27.40±0.28	25.18±0.21
Sample Generation	Zhang et al. (2021)	35.78±1.36	22.75±0.98	11.88±1.02	11.01±0.61
	Shen et al. (2021)	56.88±0.42	31.59±0.42	30.20±0.69	30.15±0.47
Ours(w/o local adaptation)		60.07±0.42	**36.11±0.56**	34.80±0.83	32.18±0.92
Ours		**61.13±0.12**	35.83±0.58	**35.21±0.48**	**33.59±0.51**

probability estimation, and the selected class is regenerated by equations (7) and (8). The $C_{N'}$ is then adjusted through the following equation.

$$W_{N'} = \underset{\tilde{W_N}}{\arg\min}\, \lambda||\tilde{W_N} - W_N||_2^2 - \Sigma_{k=1}^{K_d} \alpha \log p(y_i^k|x_i^k;\tilde{W_N}) \tag{10}$$

where λ is a hyperparameter, α_k is the weight of the k-th retrieved example and $\Sigma_{k=1}^{K_d}\alpha = 1$. We assume that all K_d retrieved examples are equally important regardless of their distance to the query vector and set $\alpha_k = \frac{1}{K}$, W_N is the weight of C_N, and $W_{N'}$ is discarded after being finetuning.

4 Experiments

4.1 Datasets and implemental setup

We conduct CIL experiment on the widely used datasets in related works (Rebuffi et al., 2017)(Shen et al., 2021), : CIFAR-10 and CIFAR-100, Tiny-ImageNet. We split CIFAR-10 dataset into 5 tasks, each task containing 2 classes. CIFAR-100 data is split into 5 tasks, and 10 tasks with 20 classes, and 10 classes per task. Split-Tiny-ImageNet is built from Tiny-ImageNet Le & Yang (2015) by splitting 200 class samples into 10 disjoint sets of samples, each consisting of 20 classes. We conduct 6-fold cross-validation for every dataset.

The network follows the original model in OWM. The encoder is composed of 3 convolutional layers with 64, 128, 256 output channels and 2×2 filter. Classifier is composed of 3 FC layers with 1000, 1000, n (class number) nodes. Our method estimates and stores one mean and covariance per class ($K = 2$) and generates 1,000 representations for each class. For Rehearsal methods, we use a memory size of 2,000 images for CIFAR10, and 20,000 images for CIFAR100, tiny-ImageNet. Note that other branches of continual learning such as parameter isolation methods, are implemented in TIL setup, so they show bad performance in CIL setup. Therefore, we excluded their results.

4.2 Results

From Table 1, our method shows the highest accuracy compared to other methods except for Joint Learning, where the network is trained with all the data as conventional deep learning. Our method achieves a performance improvement of approximately 3%p compared to the previous one in CIFAR-10 and 5%p in CIFAR-100. Tiny ImageNet the improvement of traditional contrast in about 3%p can see the effect. This shows the utility value of data presentation and proves that the data represented by multivariate performs better than before. In addition, we demonstrated the

Table 2: Accuracy on different number of Gaussian Distribution Number (K) in CIFAR10

# Distribution (K)	1	2	3	4	5	10
Accuracy (%)	60.07±0.42	62.66±0.24	63.67±0.36	63.55±0.15	64.11±0.08	64.64±0.12

Table 3: Accuracy on different number of retrive class K_d with 200 sample in CIFAR10

# retrive class(K_d)	2	3	4	5	6	7
Accuracy (%)	58.15±0.21	59.91±0.30	60.81±0.32	61.13 ± 0.12	61.08 ± 0.09	61.10±0.07

value of local adaptation through slightly better performance than without local adaptation when fine-tuning through local adaptation. Figure 4 shows the task-wise results to visualize the accuracy drop. We show that our proposed model (applied with local adaptation) does not forget more than other models and prevents CF according to the task.

Table 2 shows the performance difference in CIFAR-10 according to the number of Gaussian distributions. It can be seen that the more sophisticated data is recovered using many distributions, the better the performance is. We also demonstrate the t-SNE result of the generated representations compared to actual representations in Figure 5. It is visible that representations generated by multivariate Gaussian distribution can precisely approximate the actual distribution. Table 3 shows the performance according to the number of classes selected during the test phase. The best performance result was achieved when there were 5 classes. Based on this, we set five retrieve classes in all experimental results.

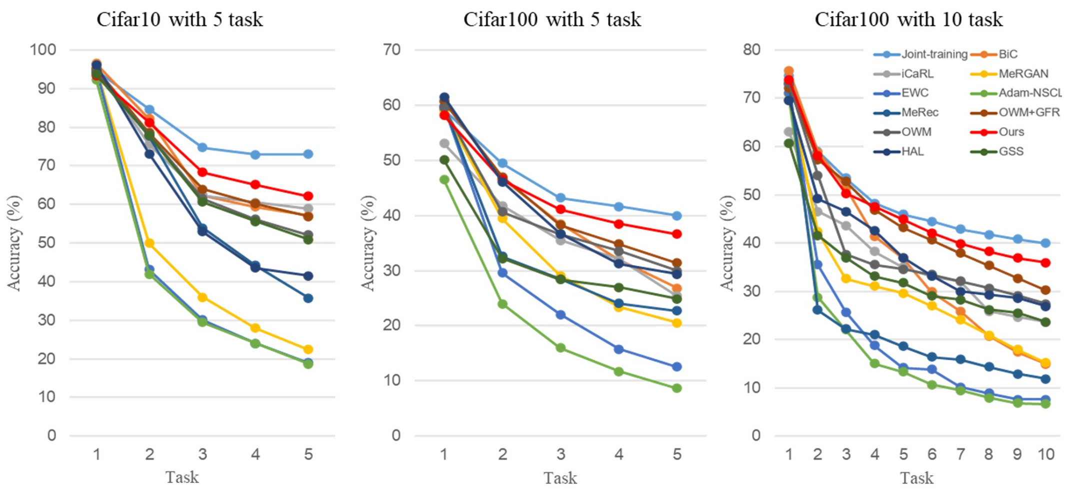


Figure 4: Task-wise accuracies (%). Joint learning shows the Upper Bound of the model.

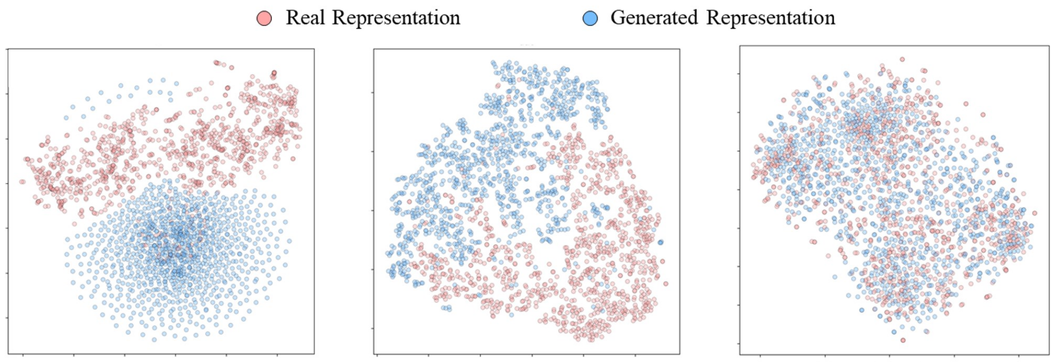
4.3 Discussion

Our method shows a bigger performance gain in longer tasks, where the performance improves 5.01%p in the 10-task CIFAR-100 experiment. This shows the scalability of our method. Kirkpatrick et al. (2017) and Wang et al. (2021) are initially implemented in TIL, so they perform poorly in this experiment. Wu et al. (2019) and Chaudhry et al. (2021) try to alleviate the class imbalance and the overfitting, but their results indicate that their methods are ineffective. In addition, our model showed about 3%p performance improvement over the previous best performance model on other datasets.

Tables 3 and 4confirmed that the appropriate retrieve class and the number of data samples are essential. When the retrieve class was small, the classification performance was low, and a good

Table 4: Accuracy on different number of sample with 5 retrive class in CIFAR10

# samples by (K_d)	50	100	150	200	300	500
Accuracy (%)	57.21±0.84	58.44±0.96	61.02±0.58	61.13 ± 0.12	60.00 ± 0.13	59.36±0.17



(a) Univariate Gaussian Distribution

(b) GAN

(c) Multivariate Gaussian Distribution

Figure 5: t-SNE of real representations and generated representations of class 0 in CIFAR10 by (a) univariate Gaussian, (b) GAN-based generator, and (c) multivariate Gaussian.

performance could be expected in more than five retrieve classes. Otherwise, the number of sampling achieved the highest performance at 200 and lower performance at 500. We think the cause of this phenomenon is that the knowledge up to T_N learned by the model is diluted when local adaptation is over-applied.

5 Conclusion

This paper points out the limitations of raw data sampling and proposes a memory-based method based on the presentation of data to solve this problem. CF was prevented by converting the presentation of data into a Gaussian distribution, storing it in memory, combining data from new tasks with data regenerated according to distributed expression, and improving performance during the test through local adaptation.

In future work, we intend to develop an inference method that prevents overfitting the test dataset by considering the local adaptation discussed in the disruption and the gradient of the existing model.

References

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019.

Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8218–8227, 2021.

Jihwan Bang, Hyunseo Koh, Seulki Park, Hwanjun Song, Jung-Woo Ha, and Jonghyun Choi. Online continual learning on a contaminated data stream with blurry task boundaries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9275–9284, 2022.

Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 233–248, 2018.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 532–547, 2018a.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018b.

Arslan Chaudhry, Albert Gordo, Puneet Dokania, Philip Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6993–7001, 2021.

Cyprien de Masson D'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. Episodic memory in lifelong language learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1): 1–22, 1977.

Jhair Gallardo, Tyler L Hayes, and Christopher Kanan. Self-supervised training enhances online continual learning. *arXiv preprint arXiv:2103.14010*, 2021.

Gunshi Gupta, Karmesh Yadav, and Liam Paull. Look-ahead meta learning for continual learning. *Advances in Neural Information Processing Systems*, 33:11588–11598, 2020.

Wenpeng Hu, Qi Qin, Mengyu Wang, Jinwen Ma, and Bing Liu. Continual learning by using information of each class holistically. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7797–7805, 2021.

Khurram Javed and Martha White. Meta-learning representations for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Chris Dongjoo Kim, Jinseo Jeong, Sangwoo Moon, and Gunhee Kim. Continual learning on noisy data streams via self-purified replay. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 537–547, 2021.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *European conference on computer vision*, pp. 524–540. Springer, 2020.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. *arXiv preprint arXiv:2103.09762*, 2021.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

Gehui Shen, Song Zhang, Xiang Chen, and Zhi-Hong Deng. Generative feature replay with orthogonal weight modification for continual learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

Pablo Sprechmann, Siddhant M Jayakumar, Jack W Rae, Alexander Pritzel, Adria Puigdomenech Badia, Benigno Uria, Oriol Vinyals, Demis Hassabis, Razvan Pascanu, and Charles Blundell. Memory-based parameter adaptation. *arXiv preprint arXiv:1802.10542*, 2018.

Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 184–193, 2021.

Chenshen Wu, Luis Herranz, Xialei Liu, Joost van de Weijer, Bogdan Raducanu, et al. Memory replay gans: Learning to generate new categories without forgetting. *Advances in Neural Information Processing Systems*, 31, 2018.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 374–382, 2019.

Jaehong Yoon, Divyam Madaan, Eunho Yang, and Sung Ju Hwang. Online coreset selection for rehearsal-based continual learning. *arXiv preprint arXiv:2106.01085*, 2021.

Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, 2019.

Baosheng Zhang, Yuchen Guo, Yipeng Li, Yuwei He, Haoqian Wang, and Qionghai Dai. Memory recall: A simple neural network training framework against catastrophic forgetting. *IEEE Transactions on Neural Networks and Learning Systems*, 33(5):2010–2022, 2021.

Song Zhang, Gehui Shen, Jinsong Huang, and Zhi-Hong Deng. Self-supervised learning aided class-incremental lifelong learning. *arXiv preprint arXiv:2006.05882*, 2020.