Entropy-based Sample Selection for Online Continual Learning

Felix Wiewel and Bin Yang
Institute of Signal Processing and System Theory
University of Stuttgart
Stuttgart, Germany
{felix.wiewel, bin.yang}@iss.uni-stuttgart.de

***Abstract*—Deep neural networks (DNNs) suffer from catastrophic forgetting, a rapid decrease in performance when trained on a sequence of tasks where only data of the most recent task is available. Most previous research has focused on the case where all data of a task is available simultaneously and boundaries between tasks are known. In this paper, we focus on the online setting where data arrives one-by-one or in small batches ordered by tasks and task boundaries are unknown. Avoiding catastrophic forgetting in such a setting is of great interest since it would allow DNNs to accumulate knowledge without the need to store all previously seen data even if task boundaries are unknown. For this, we propose a novel rehearsal algorithm for online continual learning that is derived from basic concepts of information theory. We demonstrate on commonly used data sets that our method can avoid catastrophic forgetting, achieve competitive results when compared with the current state-of-the-art and even outperform it in most cases.**

***Index Terms*—Online Continual Learning, Entropy, Rehearsal**

I. Introduction

Although DNNs have achieved remarkable success and even reached or surpassed human performance in many areas of machine learning [1], they are trained in a way that differs significantly from how humans learn [2]. While training DNNs with the backpropagation algorithm [3] requires a large data set with millions or even billions of examples that are presented to the network in a random order, humans can learn efficiently from only few examples that are usually not encountered in a random order. Training a DNN on a sequence of tasks, where only data of the most recent task is available, is one example of a setting with non-random order of the input sequence. In such a setting, DNNs typically suffer from a phenomenon called *catastrophic forgetting* [4], [5], a rapid decrease in performance on all but the most recent task. Although this phenomenon has been known since the early 1990s, it has only recently seen renewed interest through the advent of a field called continual learning. Avoiding catastrophic forgetting is of great interest since it would enable the training of DNNs in an incremental way where all data of previously learned tasks does not have to be rehearsed nor saved. The methods proposed for overcoming catastrophic forgetting can be broadly classified in three categories.

Regularization-based methods encode information about previously learned tasks in the form of a suitable regularization term in order to prevent catastrophic forgetting. *Elastic Weight Consolidation* (EWC) [6] and *Synaptic Intelligence* (SI) [7] penalize deviations from parameters that are important for the DNNs performance on previous tasks during training on a new task. EWC uses a diagonal approximation to the Fischer Information matrix of the log-likelihood parameterized by the neural network to determine the importance of a parameter. SI, on the other hand, estimates a parameters importance based on its trajectory through parameter space during the training process.

Structural methods use a combination of freezing weights and expanding the architecture of a DNN in order to mitigate catastrophic forgetting. While *Progressive Neural Networks* (PNN) [8] freezes a DNN's weights and expands its layers for every new task, *Dynamically Expandable Networks* (DEN) [9] uses a combination of selective retraining, dynamic network expansion and splitting to efficiently adapt a DNN to new tasks without affecting its performance on previously learned ones.

Rehearsal methods utilize some sort of memory, which stores previously encountered training examples, in order to prevent catastrophic forgetting. *Gradient Episodic Memory* (GEM) [10] uses training examples from previously learned tasks to force gradient descent steps during training of a new task in a direction that does not increase the loss on them. In *Deep Generative Replay* (DGR) [11] a Generative Adversarial Network (GAN) [12] is used to implicitly store previous examples using a generative model. During training of a new task, data sampled from the GAN is mixed with new data in a suitable proportion in order to avoid catastrophic forgetting.

While the above methods differ quite a lot with respect to how they try to prevent catastrophic forgetting, they have in common that the boundaries between different tasks during training have to be known, i.e. the methods rely on a clear and known transition from one task to another. This prevents their application to online settings where training examples arrive in small batches or even one at a time and boundaries between tasks are unclear. Recently, Aljundi et al. proposed and evaluated a rehearsal based method called *Gradient based Sample Selection* (GSS) for such an online setting in [13]. It relies on gradient information in order to maintain a buffer of diverse samples for rehearsal. Inspired by this work, we propose and evaluate a novel method for online continual learning derived from basic concepts of information theory and demonstrate that our method achieves competitive results and even outperforms GSS on most benchmarks.

II. Problem Formulation

We consider a variation of the incremental class learning setup as described in [14] where a DNN is presented with a series of N classification tasks, each represented by a set of input-output pairs, i.e. $\mathcal{T}_{1\leq i\leq N} = \{\mathbf{x}_j^i, y_j^i | 1 \leq j \leq M_i\}$. The j-th input of the i-th task $\mathbf{x}_j^i \in \mathbb{R}^S$ is represented by a real-valued vector, e.g. the pixels of an image, and the corresponding output $y_j^i \in \mathbb{Z}$ is an integer number. Similar to [13] the input-output pairs are observed as an ordered sequence, like $\mathbf{x}_1^1, y_1^1; \ldots; \mathbf{x}_{M_1}^1, y_{M_1}^1; \ldots; \mathbf{x}_1^N, y_1^N; \ldots; \mathbf{x}_{M_N}^N, y_{M_N}^N$. It is important to note that in contrast to most previous work on continual learning, we do not assume knowledge about task boundaries and each sample is only observed once. We are then interested in learning the parameters $\boldsymbol{\theta}$ of a DNN $f_{\boldsymbol{\theta}}$ such that the empirical risk [15] is minimized. Mathematically, this can be written as

$$\boldsymbol{\theta} = \arg\min_{\tilde{\boldsymbol{\theta}}} \sum_k \mathcal{L}\left(f_{\tilde{\boldsymbol{\theta}}}\left(\mathbf{x}_k\right), y_k\right), \tag{1}$$

where $\mathcal{L}$ is a suitable loss function, like the cross entropy loss for classification, and $\mathbf{x}_k, y_k \in \bigcup_{i=1}^N \mathcal{T}_i$ are training examples from all tasks. This setup is much more challenging than traditional continual learning, since input-output pairs are observed only once and task boundaries are unknown, which makes it difficult to find $\boldsymbol{\theta}$ if we are not able or willing to store all data.

III. Proposed Method

To still train a DNN in an online setting as described in Section II, a buffer $\mathcal{B} = \{\mathbf{x}_k, y_k\}_{k=1}^L$ that stores L previous inputs $\mathbf{x}_k$ and their labels y_k is used. This buffer is used for rehearsal on previously seen examples. As an individual input-output pair $\mathbf{x}, y$ arrives, a suitable algorithm decides if it joins the buffer or not and if it does, which old example in the buffer is replaced. Then the newly arrived data from the input sequence is collected in a minibatch, combined with randomly sampled data from the buffer and used to update the DNNs parameters using stochastic gradient descent. For a method to successfully avoid catastrophic forgetting it is important that the algorithm for managing the buffer keeps diverse and representative examples of previously learned classes. In this paper, we adapt an information theoretic perspective on the process of managing a buffer and use basic principles to derive our method.

We start by considering the process of selecting an input-output pair $\mathbf{x}, y$ from our buffer for rehearsal as sampling from a joint distribution $P(\mathbf{X}, Y) : \mathbb{R}^S \times \mathbb{Z} \rightarrow [0, 1]$ of the random variables $\mathbf{X}$ and Y. Note that $P(\mathbf{X}, Y)$ is not the joint data distribution of the input sequence, rather an approximation of it based on the samples stored in the buffer. Suppose we fill our buffer with L copies of the same input-output pair $\mathbf{x}, y$. This would certainly not be diverse and representative, since only one example of just one class is used for rehearsal. We can then evaluate the well known joint Shannon entropy [16] of $\mathbf{X}$ and Y given by

$$H(\mathbf{X}, Y) = -\mathbb{E}\left[\log P(\mathbf{X}, Y)\right], \tag{2}$$

where the expectation is taken with respect to $P(\mathbf{X}, Y)$. Since we always sample the same input-output pair $\mathbf{x}, y$ from the buffer, i.e. $P(\mathbf{x}, y) = 1$, the resulting entropy and therefore the informational value when sampling from the distribution will be zero. The basic idea of our method is therefore to add only those samples to the buffer that are likely to increase the entropy. Although this idea is similar to maximum entropy sampling [17] or the training of decision trees [18], it has not been applied to online continual learning.

Note that if we add another input-output pair $\mathbf{x}, y$ to our buffer and then sample from it, we are essentially sampling from a different distribution $P(\mathbf{X}, Y)$. In order to derive an algorithm that adds only those examples to our buffer that are likely to increase the joint entropy of the random variables $\mathbf{X}$ and Y if we sample from it, we rewrite the joint entropy as

$$\begin{aligned} H(\mathbf{X}, Y) &= -\mathbb{E}\left[\log P(\mathbf{X}|Y)\right] - \mathbb{E}\left[\log P(Y)\right] \\ &= H(\mathbf{X}|Y) + H(Y). \end{aligned} \tag{3}$$

This shows that we can evaluate the entropy using two independent terms where one only depends on the marginal $P(Y)$ and the other on the conditional distribution $P(\mathbf{X}|Y)$. Since Y is a discrete random variable used to model the integer labels of our examples, it follows a categorical distribution. We estimate the individual class probabilities by determining the relative frequency of the labels stored in our buffer. Unfortunately we do not know the type of distribution of $P(\mathbf{X}|Y)$. Therefore we use *Kernel Density Estimation* (KDE) [19] to estimate $P(\mathbf{X}|Y)$ as

$$P(\mathbf{x}|y) \approx \frac{1}{M_y} \sum_{k=1}^{M_y} K(\mathbf{x} - \mathbf{x}_y\left[k\right]), \tag{4}$$

where M_y is the number of examples $\mathbf{x}_y$ in the buffer with label y and $K : \mathbb{R}^S \rightarrow [0, 1]$ is a kernel function. With these approximations we can now derive an algorithm that tries to maximize the joint entropy $H(\mathbf{X}, Y)$.

If an individual input-output pair $\mathbf{x}, y$ arrives, it is added to the buffer until it is full. For every example that arrives after the buffer is filled we have to decide if it replaces a stored example or not. If it does, we also have to decide which one it replaces. When using rehearsal in combination with a small buffer, there is always a risk of overfitting, i.e. the training error is very small while the generalization to unseen data is bad. In order to avoid this, we will always add a new sample to the buffer even if this causes a small decrease in entropy. This means that we are not trying to strictly maximize the entropy when adding a sample to the buffer but accept small variations of it in order to avoid overfitting. For this, we first try to maximize $H(Y)$. Given that $P(Y)$ is a categorical distribution, it is well known that the entropy of it is at its maximum if the individual class probabilities are identical. This means that if the relative frequency of the labels in our

Algorithm 1 Entropy-based Sample Selection

```
Require: $\mathbf{x}$, $y$, Buffer $\mathcal{B}$
Ensure: Updated $\mathcal{B}$
 1: if $|\mathcal{B}| < L$ then
 2:     $\mathcal{B} \leftarrow \mathbf{x}, y$
 3: else
 4:     $\mathcal{C} \leftarrow$ samples of majority class in $\mathcal{B}$
 5:     for all $\mathbf{x}_i \in \mathcal{C}$ do
 6:         $d_i = \min_{\mathbf{x}_j \in \mathcal{C}} \|\mathbf{x}_i - \mathbf{x}_j\|_2$
 7:     end for
 8:     $i \sim P(i) = (1 - d_i)/\sum_j (1 - d_j)$
 9:     $\mathbf{x}_i, y_i \leftarrow \mathbf{x}, y$
10: end if
```

buffer are not identical, we can increase the entropy $H(\mathbf{X}, Y)$ by first replacing an example from the majority class by one of another class. In the case that an example from the majority class arrives, we use it to replace an example in the buffer that is from the same class. This does not change $H(Y)$ but it might decrease $H(\mathbf{X}|Y)$. We therefore try to minimize a possible decrease in $H(\mathbf{X}, Y)$ by choosing an a sample from the buffer for replacement that maximizes $H(\mathbf{X}|Y)$ in this case.

With the knowledge that always replacing an example of the majority class in the buffer does not decrease $H(Y)$, we now have to decide which one we have to replace in order to maximize the conditional entropy. For this, we use the KDE from Eq. (4) to substitute $P(\mathbf{X}|Y)$ in $H(\mathbf{X}|Y)$ yielding

$$H(\mathbf{X}|Y) \approx -\mathbb{E}\left[\log \frac{1}{M_y} \sum_{k=1}^{M_y} K(\mathbf{x} - \mathbf{x}_y[k])\right], \quad (5)$$

where we approximate the expectation over $\mathbf{x}$ by a summation over all examples in our buffer. This means for every new input-output pair $\mathbf{x}, y$ in our buffer, we compute its distance to all other examples $\mathbf{x}_y[k]$ of the same class in the buffer, apply the kernel function and take the average. We repeat this process for every sample in the buffer. We now assume that the kernel function $K(\mathbf{x})$ is isotropic and has infinite support. A typical example of such a kernel is the Gaussian, i.e. $K(\mathbf{x}) = (2\pi)^{-1/2} \exp(-\|\mathbf{x}\|_2^2/2)$. Since the kernel function is decreasing with the same rate in all directions as $\|\mathbf{x}\|_2^2$ increases, maximizing the expectation in Eq. (5) is equivalent to maximizing the minimum distance between all examples of every class in the buffer.

A simple approach for this is to randomly replace an example of the majority class that is close to another one of its class with high probability. For this, we choose the i-th example with probability $(1 - d_i)/\sum_j(1 - d_j)$, where d_i is the minimum distance of $\mathbf{x}_i$ to all examples of the same class in the buffer. This does not always increase the minimum distance between all examples of every class in the buffer. For example, if we replace a sample with another one that has a smaller minimum distance to examples of the same class stored in the buffer, our approximation of the expectation in Eq. (5) will be increased. But our experiments show that this is a fast and effective way of selecting a sample for replacement. Additionally the random nature of this approach helps in avoiding a static buffer and therefore the risk of overfitting when using it for rehearsal.

TABLE I
RESULTS ON THE CIFAR10 DATA SET WITH A BUFFER SIZE OF 500.

# first task	2000	4000	400
# other tasks	2000	400	4000
Method	Mean Acc. ± std. [%]		
Ours	36.76 ± 0.83	$\mathbf{35.00 \pm 1.43}$	35.27 ± 0.99
GSS [13]	36.67 ± 0.96	28.41 ± 2.13	$\mathbf{36.66 \pm 1.00}$
Reservoir	$\mathbf{37.12 \pm 1.30}$	25.64 ± 1.52	34.83 ± 0.57
Random	16.43 ± 0.28	23.39 ± 1.35	16.31 ± 0.27

Overall we can summarize our method as follows. When a new input-output pair $\mathbf{x}, y$ arrives we add it to the buffer until it is full. If this is the case, we then use the relative frequency of the classes in our buffer to determine the majority class. Afterwards we choose an example of this class for replacement where the probability of a particular example to be replaced is high if the minimum distance to all other examples of the same class in the buffer is small. This example is then replaced with the newly arrived one. We choose this approach over strictly maximizing the entropy, since it helps in avoiding overfitting when the buffer is used for rehearsal. Our method is summarized in Algorithm 1.

IV. Experiments

We evaluate our method on two data sets that are commonly used in the continual learning literature, MNIST [20] and CIFAR10 [21]. While MNIST features 60000 training and 10000 test images of handwritten digits with a size of $28 \times 28 \times 1$ pixel, CIFAR10 contains 50000 training and 10000 test images of various objects from 10 classes with a size of $32 \times 32 \times 3$ pixel. For both data sets we use a common setup. We first split the data sets into five tasks where each task contains only training examples from two classes. We choose those in an increasing order, i.e. $0, 1 \rightarrow \ldots \rightarrow 8, 9$, and concatenate all examples from these tasks in one sequence. During training we expose the model to mini batches from this sequence, such that input-output pairs are presented to the model only once. This effectively simulates an online setting where examples arrive in small batches or even individually. After we have trained the model on such a sequence, we evaluate its performance on the complete test set. All results are averaged over ten runs. Since in practice it is not unusual for data sets to be unbalanced, we also simulate such a scenario by reducing/increasing the number of examples from the first task when compared with the other tasks. We could also reduce/increase the number of examples from every other task but this is the most challenging scenario for our method, because it has to maintain a representative buffer for the minority/majority classes for the maximum amount of time. Similar to [13], we compare our method with the naive approach of randomly selecting L samples from the concatenation of a newly arriving batch with

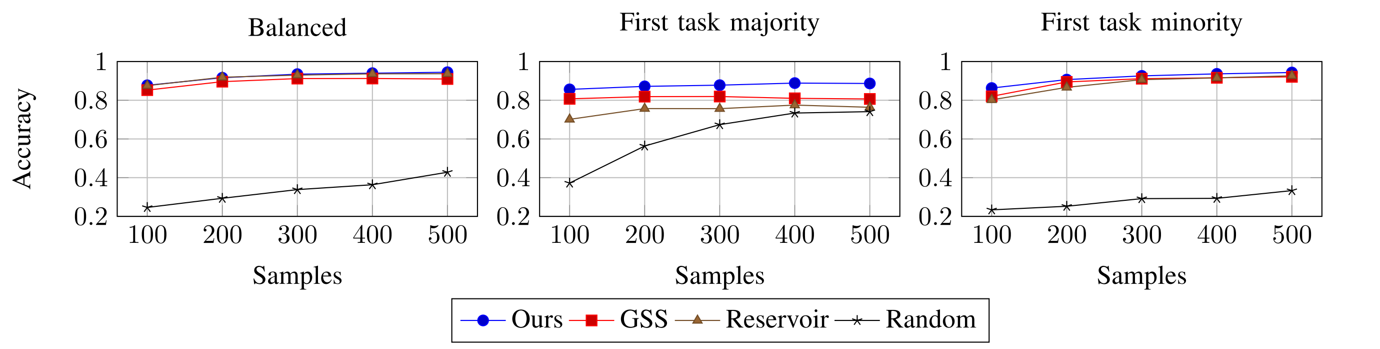

Fig. 1. Comparison with reservoir sampling base line and Gradient Based Sample Selection (GSS) [13]. Mean accuracy over ten runs on the disjoint MNIST data set for different buffer sizes. The plot shows the results when each task is represented by 1000 examples (left), for the first task having 2000 while all other tasks have 200 examples (middle) and the first task being represented by 200 while all other tasks have 2000 examples (right).

the replay buffer and the much stronger base line of reservoir sampling [22], where the n-th sample in the sequence replaces a randomly chosen one from the buffer with probability $1/n$.

A. Architecture, Hyperparameters and Training Protocol

For all experiments on the MNIST and CIFAR10 data sets we use the same architecture and hyperparameters. It consists of three *ReLU* activated convolutional layers with 32, 64, and 128 filters. Each of these is followed by a max-pooling layer with pooling size $(2, 2)$. The last layer is a dense layer with ten neurons and Softmax activation function. The *SGD* optimizer with a batch size of 32 and a learning rate of 0.001 is used for all experiments. During training on the sequence of input-output pairs, we perform one gradient descent step on each, a batch containing only new data and ten batches of data sampled from the buffer for rehearsal. We do this since training only on examples that are sampled from the buffer can lead to overfitting if its size is small and the examples in it are not replaced regularly.

B. Disjoint MNIST

For our experiments on the MNIST data set we split the data set according to Section IV into five tasks represented by disjoint sets containing only examples of two classes. Similar to [13] we evaluate our method on this sequence not only for the case where all tasks have the same number of 1000 examples but also in cases where the first task is under- or overrepresented. For the former we use a sequence with 200 examples of the first task and 2000 from every other task while for the latter we use a sequence with 2000 examples of the first task and 200 from every other task. The results for different buffer sizes are averaged over ten runs and summarized in Fig. 1. As can be seen, our method not only outperforms the random and reservoir sampling base lines but also GSS. It is interesting to note that GSS seems to yield slightly worse results than our method and reservoir sampling in the case of a balanced number of examples per task. Our method, on the other hand, always outperforms reservoir sampling in this experiment. We attribute this to the regular replacement of samples in the buffer when using our method. If the first task is represented by a majority of examples, GSS achieves significantly better results than random and reservoir sampling. Here, our method outperforms all other methods by a significant margin. This experiment is challenging for all methods since they have to quickly build a representative buffer after a transition to new tasks. If the first task is represented by a minority of examples the results are almost identical to the balanced case. The only notable difference is that now GSS outperforms reservoir sampling and the performance of all methods is slightly decreased for a small buffer size.

C. Disjoint CIFAR10

For our experiments on the CIFAR10 data set, which is much more difficult than the MNIST data set, we also evaluate our method on three sequences but with double the number of examples for every task. We only report results for a buffer size of 500 that are summarized in TABLE I. When the number of examples per task is balanced, our method and GSS achieve nearly identical results while reservoir sampling is slightly better. But although the mean accuracy of reservoir sampling is slightly higher than that of our method, its standard deviation is significantly higher. Much more interesting results can be observed when the first task is represented by a majority of the examples. In this case our method outperforms all other methods by a significant margin while the performance difference between GSS, reservoir and random sampling is small but noticeable. When the first task is represented by a minority of the examples, however, GSS performs best. But even in this case our method achieves only slightly worse results that are still better than reservoir sampling. As one would expect, the naive approach of random sampling achieves underwhelming results in every experiment.

D. Visualization

In order to verify if our method according to Algorithm 1 actually maximizes the entropy when sampling from the buffer, we estimate the joint entropy $H(\mathbf{X}, Y)$ according to Section III during experiments using a Gaussian kernel. For this we choose the sequence from Section IV where the first tasks is represented by 2000/4000 and the other tasks by 200/400 examples. We then estimate $H(Y)$ using the relative

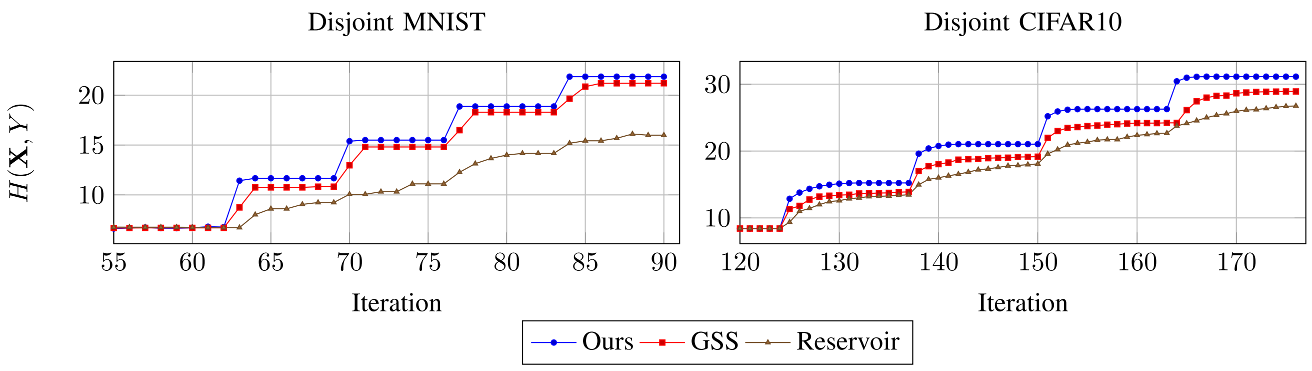

Fig. 2. The estimated entropy $H(\mathbf{X}, Y)$ over the iterations during training for different methods using a buffer size of 100 for disjoint MNIST and 500 for disjoint CIFAR10. The plot shows the entropy over one training run from the experiment explained in Sections IV where the first task is represented by 2000/4000 and all other tasks by 200/400 examples.

frequency of the class labels in the buffer and $H(\mathbf{X}|Y)$ by sampling using the KDE from Eq. (4). Since we know that the modes of $P(\mathbf{X}|Y)$ and therefore most of the probability mass is located near $\mathbf{x}_y[k]$, we estimate the expectation by sampling at these locations. The results are summarized in Fig. 2. As can be seen, our method consistently achieves a higher entropy than all other methods. Although GSS is not motivated by maximizing the entropy, it still maintains a buffer of diverse examples and therefore a quite high entropy. It is also interesting to note that task boundaries are clearly visible, since both our method and GSS are able to significantly increase the estimated entropy of their buffer. We excluded random sampling from this plot, since it is only able to keep a buffer with examples from the most recent task. When we sample from it, we are therefore sampling from a distribution over only two and not over all previously seen classes. This makes a comparison between the entropy of random sampling and the other methods uninformative.

V. Conclusion

In this paper, we propose a novel method for online continual learning that avoids catastrophic forgetting. We derive our method by considering the process of sampling from a buffer for rehearsal from an information theoretic perspective. Using the Shannon entropy as a measure for the informational value of this process, we derive an algorithm that tries to maximize it and therefore keeps a buffer that is diverse and representative of previously seen classes. On commonly used benchmarks, our method outperforms the state-of-the-art in most cases.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.

[2] Y. Bengio, D. Lee, J. Bornschein, and Z. Lin, “Towards biologically plausible deep learning,” *CoRR*, vol. abs/1502.04156, 2015. [Online]. Available: http://arxiv.org/abs/1502.04156

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[4] S. Lewandowsky and S.-C. Li, “Catastrophic interference in neural networks: causes, solutions, and data,” in *Interference and inhibition in cognition*. Elsevier, 1995, pp. 329–361.

[5] R. M. French, “Catastrophic forgetting in connectionist networks,” *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[6] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the national academy of sciences*, p. 201611835, 2017.

[7] F. Zenke, B. Poole, and S. Ganguli, “Continual learning through synaptic intelligence,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3987–3995.

[8] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, “Progressive neural networks,” *CoRR*, vol. abs/1606.04671, 2016. [Online]. Available: http://arxiv.org/abs/1606.04671

[9] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, “Lifelong learning with dynamically expandable networks,” in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=Sk7KsfW0-

[10] D. Lopez-Paz *et al.*, “Gradient episodic memory for continual learning,” in *Advances in Neural Information Processing Systems*, 2017, pp. 6467–6476.

[11] 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.

[12] 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 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[13] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, “Gradient based sample selection for online continual learning,” in *Advances in Neural Information Processing Systems*, 2019, pp. 11 816–11 825.

[14] Y. Hsu, Y. Liu, and Z. Kira, “Re-evaluating continual learning scenarios: A categorization and case for strong baselines,” in *Continual Learning Workshop NeurIPS*, 2018.

[15] V. Vapnik, “Principles of risk minimization for learning theory,” in *Advances in neural information processing systems*, 1992, pp. 831–838.

[16] C. E. Shannon, “A mathematical theory of communication,” *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[17] M. C. Shewry and H. P. Wynn, “Maximum entropy sampling,” *Journal of applied statistics*, vol. 14, no. 2, pp. 165–170, 1987.

[18] J. R. Quinlan, “Induction of decision trees,” *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[19] R. A. Davis, K.-S. Lii, and D. N. Politis, “Remarks on some nonparametric estimates of a density function,” in *Selected Works of Murray Rosenblatt*. Springer, 2011, pp. 95–100.

[20] Y. LeCun, C. Cortes, and C. Burges, “Mnist handwritten digit database,” *ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, 2010.

[21] A. Krizhevsky, “Learning multiple layers of features from tiny images,” Tech. Rep., 2009.

[22] J. S. Vitter, “Random sampling with a reservoir,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 11, no. 1, pp. 37–57, 1985.