CVPR 2023 Continual Learning Competition——Class-incremental with Repetition

1st Shuai Xue *Computer Science and Technology*
Northwestern Polytechnical University
Xi'an, China
1992740840@qqcom
2nd Yuanzhuo Niu *Computer Science*
Northwestern Polytechnical University
Xi'an, China
1773548849@qq.com

***Abstract*—Currently, benchmarks for continual learning often use a very specific type of stream, in which each experience is only seen once and with no overlap between the experiences. Although these benchmarks have proven useful for academic purposes, they do not reflect the arbitrary non-stationarity that can be observed in the real world. In this challenge, the goal is to design efficient strategies for a class of continual learning problems we refer to as Class-incremental with Repetition (CIR). We randomly switched several plugins in the official code, and handwrote data replay for dealing with catastrophic forgetting. What's more, we employ hard attention to the task with two repeat technique: fragments and ensemble. Experimental results deploy that our last model has the most outstanding performance than the others**

***Index Terms*—Continual Learning, Class-incremental with Repetition, HAT, avalanche**

I. Introduction

Continual Learning (CL) requires a model to learn new information from a stream of experiences presented over time, without forgetting previous knowledge . The nature and characteristics of the data stream can vary a lot depending on the real-world environment and target application. The Class-Incremental (CI) scenario is the most popular one in CL. CI requires the model to solve a classification problem where new classes appear over time. Importantly, when a set of new classes appears, the previous ones are never seen again. However, the model still needs to correctly predict them at test time. Conversely, in a Domain-Incremental (DI) scenario the model sees all the classes at the beginning and continues to observe new instances of the classes over time.

The CI and DI scenarios have been very helpful to promote and drive CL research in the last few years. However, they strongly constrain the properties of the data stream in a way that is sometimes considered unrealistic or very limiting . Recently, the idea of Class-Incremental with Repetition (CIR) scenario has started to gather some attention in CL. The CIR scenario is arguably more flexible in the definition of the stream since they allow both the introduction of new classes and the repetition of previously seen classes. Crucially, repetition is a property of the environment and cannot be controlled by the CL agent. This is very different from Replay strategies, where the repetition of previous concepts is heavily structured and can be tuned at will.

The CIR [1] question we discuss comes from the 4th CLVISION CVPR Workshop challenge.In that challenge, the goal is to design efficient strategies for a class of continual learning problems we refer to as Class-incremental with Repetition (CIR). CIR encompasses a variety of streams with two key characteristics: (i) previously observed classes can reappear in a new experience with arbitrary repetition patterns, and (ii) not all classes have to appear in every experience. Since many existing strategies were developed for continual learning problems without repetition, it is unclear how they would perform and compare in CIR streams. To explore the significance of repetition and its relevance for developing novel strategies, we provide a set of CIR benchmarks created by a stream generator that is controlled by three parameters with clear interpretation. Participants are asked to develop strategies that, after the model has finished training on the entire stream, achieve high average accuracy on a test set that contains an equal number of unseen examples of all classes in the stream.

So far, researchers have conducted considerable research on continuous learning. They have proposed continuous learning algorithms based on experience replay, network expansion, and loss regularization. However, since the original intention of these algorithms is traditional class incremental and domain incremental learning, it is difficult to adapt to the CIR data flow model proposed in the competition. So, in this article, we will introduce our own continuous learning algorithm based on experience replay, and the optimized version of the HAT [2] algorithm proposed by the winner of the competition.Our main work is as follows:

1) Some basic continuous learning algorithms, such as LwF [3] and EWC [4], were tested on the CIR data stream.
2) The experience replay algorithm for this CIR data stream was implemented under the avalanche framework, and achieved better results than the basic continuous learning algorithm (but this is somewhat cheating).

3) We re-ran the competition winner's code and expected to learn relevant algorithm details and programming ideas.

II. Related work

Current CL methods are mainly focused on two types of benchmarks, namely, Multi Task (MT) and Single Incremental Task (SIT). MT divides training data into distinct tasks and labels them during training and inference. SIT splits a single task into a sequence of unlabeled experiences. SIT can be further divided into Domain-Incremental (DI) where all classes are seen in each experience, and Class-Incremental (CL) where each experience contains only new (unseen) classes. Both DI and CI are extreme cases and are unlikely to hold in real-world environments. In a more realistic setting, the role of natural repetition in CL streams was studied in the context of New Instances and Classes (NIC) scenario and the CRIB benchmark. NIC mainly focuses on small experiences composed of images of the same object, and repetitions in CRIB are adapted to a certain dataset and protocol. The Class-Incremental with Repetition (CIR) scenario was initially formalized in , however, the work lacks a systematic study of CIR streams as the wide range of CIR streams makes them difficult to study.

A. Expansion and Interaction for Class-incremental Learning with Repetition

Zhilin Zhu from Tianjin University propose a parameter isolation method combined with an ensemble learning strategy. To achieve better model integration, they align the classifiers to maintain the same energy level obtained by the model in different training phases. To address the issue of sample efficiency, they use self-supervised learning to capture more general and discriminative representations, thus improving generalization performance. Additionally, they establish a shared prompt pool to facilitate interaction between different tasks and categories, promoting knowledge fusion.

B. Continual Learning via Feature Extractors

Benedikt Tscheschner from Graz University of Technology proposes their method.The core idea is to learn an ensemble of Feature Extractors (FE) on selected experiences, which should provide robust features useful for discriminating downstream classes. they have some heuristics that decide at each experience if they want to learn a new FE for the current classes. We do not consider experiences with less than 5 classes, stop adding FEs to the ensemble after they have seen 85% of the classes, and always train an FE on the first experience. they learn them with both a cross-entropy head, and a contrastive loss with emphasis on the hard negative pairs on a separate head. Both losses are balanced with an adaptive alpha, automatically computed depending on the energy of each loss. After learning from the current experience, the heads are removed and the backbone is frozen and added to the rest of the ensemble. However, since none of the FEs has knowledge of each other, they need to align their representations for the unified classification by using pseudo-feature projection. Therefore, regardless of a FE being trained, they always update the unified head trained on all ensemble representations. To balance the unified head, they draw inspiration from FeTrIL, extending the pseudo-feature projection to estimate also with the standard deviation and with the outputs of all the ensemble. When the mean and standard deviation of some class is not available, they replace them by re-using the current experience representations.

C. Dynamically Architecture Based on Gated Units for Class-Incremental Learning

YuXiang Zheng from Nanjing University of Aeronautics and Astronautics proposed their method which is a dynamic architecture based on gated networks. They set up independent branches for each experience and control whether the current branch is activated through gating units. During training, the gating unit opens the branch for the current experience, allowing the network to learn, while closing the branches for previous experiences to freeze their parameters. To improve the model's generalization and robustness during training, they use a large number of data augmentation techniques, such as augmix. During testing, the gating unit controls the branches to perform sequential predictions. However, it is important to note that considering all branch predictions simultaneously, as in DER, may not yield good results because most branches may not have seen this class and may make overconfident judgments. This problem can be seen as an open-set recognition problem. To address this issue, they propose a weighted strategy based on entropy, feature norm, and the number of classes. Specifically, for each branch, they calculate the entropy of the predicted probability. If the entropy is high, the sample is likely an open-set sample for the current branch. Similarly, they also calculate the feature norm. If the feature norm is high, the sample is also likely an open-set sample. Finally, they believe that the more classes a experience has, the more reliable the model's judgment will be. Therefore, they also weight based on the number of classes in the experience.

D. Overcoming Catastrophic Forgetting with Hard Attention to the Task

Catastrophic forgetting occurs when a neural network loses the information learned in a previous task after training on subsequent tasks. This problem remains a hurdle for artificial intelligence systems with sequential learning capabilities. Joan Serra propose a task-based hard attention mechanism that preserves previous tasks' information without affecting the current task's learning. A hard attention mask is learned concurrently to every task, through stochastic gradient descent, and previous masks are exploited to condition such learning. The attention vectors of previous tasks are used to define a mask and constrain the updates of the network's weights on current tasks. Since masks are almost binary, a portion of the weights remains static while the rest adapt to the new task.Their task-level hard attention mechanism, learn the Hard attention mask through SGD, and use the previous mask to adjust the learning process. This can effectively reduce catastrophic forgetting. At the same time, it is very robust

to different hyperparameters and provides a lot of adjustment possibilities.

III. METHOD

In this section, we mainly introduce the algorithm framework of the winner in the competition.

A. Hard Attention Mechanism

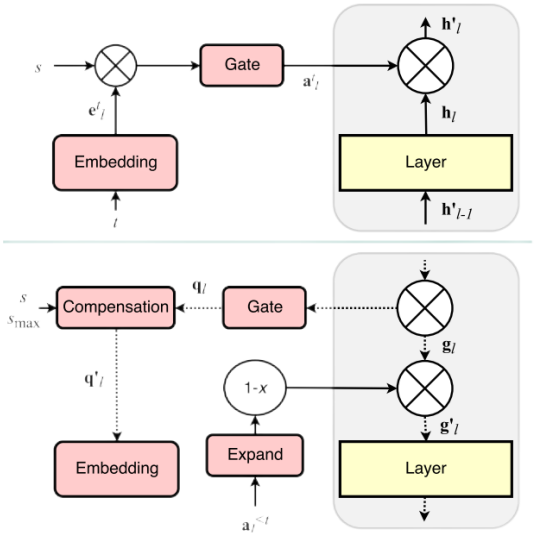


Fig. 1: Schematic diagram of the proposed approach: forward (top) and backward (bottom) passes.

To condition to the current task t, we employ a layer-wise attention mechanism figure 1. Given the output of the units2 of layer l, $\mathbf{h}_l$,we element-wise multiply $\mathbf{h}_l^{'} = \mathbf{a}_l^t \bigodot \mathbf{h}_l$.However, an important difference with common attention mechanisms is that, instead of forming a probability distribution, $\mathbf{a}_l^t$ is a gated version of a single-layer task embedding $\mathbf{e}_l^t$,

$$\mathbf{a}_l^t = \sigma(s\mathbf{e}_l^t) \tag{1}$$

where $\sigma(x) \in [0, 1]$is a gate function and s is a positive scaling parameter. We use a sigmoid gate in our experiments, but note that other gating mechanisms could be used. All layers $l = 1, ..., L-1$ operate equally except the last one, layer$L$, where a_L^t is binary hard-coded. The operation of layer L is equivalent to a multi-head output , which is routinely employed in the context of catastrophic forgetting.

The idea behind the gating mechanism of Eq 1 is to form hard, possibly binary attention masks which, act as "inhibitory synapses", and can thus activate or deactivate the output of the units of every layer. In this way, and similar to PathNet , we dynamically create and destroy paths across layers that can be later preserved when learning a new task. However, unlike PathNet, the paths in HAT are not based on modules, but on single units. Therefore, we do not need to pre-assign a module size nor to set a maximum number of modules per task. Given some network architecture, HAT learns and automatically dimensions individual-unit paths, which ultimately affect individual layer weights. Furthermore, instead of learning paths in a separate stage using genetic algorithms, HAT learns them together with the rest of the network, using backpropagation and SGD.

To preserve the information learned in previous tasks upon learning a new task, we condition the gradients according to the cumulative attention from all the previous tasks. To obtain a cumulative attention vector, after learning task t and obtaining $\mathbf{a}_l^t$, we recursively compute

$$\mathbf{a}_l^{\leq t} = \max\left(\mathbf{a}_l^t, \mathbf{a}_l^{\leq t-1}\right)$$

, using element-wise maximum and the all-zero vector for $\mathbf{a}_l^{\leq 0}$.This preserves the attention values for units that were important for previous tasks, allowing them to condition the training of future tasks.

To condition the training of task $t+1$, we modify the gradient $g_{l,ij}$ at layer l with the reverse of the minimum of the cumulative attention in the current and previous layers:

$$g'_{l,ij} = \left[1 - \min\left(a_{l,i}^{\leq t}, a_{l-1,j}^{\leq t}\right)\right] g_{l,ij}, \tag{2}$$

where the unit indices i and j correspond to the output (l) and input $(l-1)$ layers, respectively. In other words, we expand the vectors $\mathbf{a}_l^{\leq t}$ and $\mathbf{a}_{l-1}^{\leq t}$ to match the dimensions of the gradient tensor of layer l, and then perform a elementwise minimum, subtraction, and multiplication (Fig. 1). We do not compute any attention over the input data if this consists of complex signals like images or audio. However, in the case such data consisted of separate or independent features, one could also consider them as the output of some layer and apply the same methodology.

Note that, with Eq. 2, we create masks to prevent large updates to the weights that were important for previous tasks. This is similar to the approach of PackNet, which was made public during the development of HAT. In PackNet, after an heuristic selection and retraining, a binary mask is found and later applied to freeze the corresponding network weights. In this regard, HAT differs from PackNet in three important aspects. Firstly, our mask is unit-based, with weight-based masks automatically derived from those. Therefore, HAT also stores and maintains a lightweight structure. Secondly, our mask is learned, instead of heuristically- or rule-driven. Therefore, HAT does not need to pre-assign compression ratios nor to determine parameter importance through a post-training step. Thirdly, our mask is not necessarily binary, allowing intermediate values between 0 and 1. This can be useful if we want to reuse weights for learning other tasks, at the expense of some forgetting, or we want to work in a more online mode, forgetting the oldest tasks to remember new ones.

B. Supervised Contrastive Learning

This method [5] is structurally similar to that used for self-supervised contrastive learning, with modifications for supervised classification. Given an input batch of data, we first apply data augmentation twice to obtain two copies of the batch. Both copies are forward propagated through the encoder network to obtain a 2048-dimensional normalized embedding. During training, this representation is further

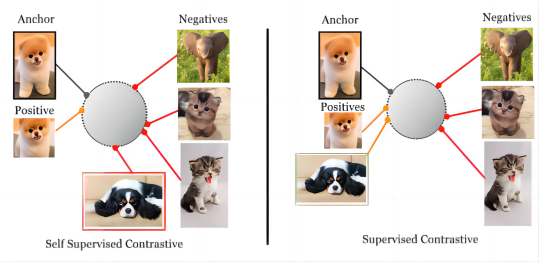


Fig. 2: Supervised vs. self-supervised contrastive losses: The self-supervised contrastive loss (left) contrasts a single positive for each anchor (i.e., an augmented version of the same image) against a set of negatives consisting of the entire remainder of the batch. The supervised contrastive loss (right) considered in this paper , however, contrasts the set of all samples from the same class as positives against the negatives from the remainder of the batch. As demonstrated by the photo of the black and white puppy, taking class label information into account results in an embedding space where elements of the same class are more closely aligned than in the self-supervised case.

propagated through a projection network that is discarded at inference time. The supervised contrastive loss is computed on the outputs of the projection network. To use the trained model for classification, we train a linear classifier on top of the frozen representations using a cross-entropy loss.

The main components of supervised contrastive learning framework are:

- *Data Augmentation module*, $Aug(\cdot)$. For each input sample, x we generate two random augmentations, $\tilde{x} = Aug(x)$, each of which represents a different $view$ of the data and contains some subset of the information in the original sample.
- *Encoder Network*,$Enc(\cdot)$, which maps x to a representation vector, $r = Enc(\boldsymbol{x}) \in \mathcal{R}^{D_E}$. Both augmented samples are separately input to the same encoder, resulting in a pair of representation vectors. r is normalized to the unit hypersphere in $\mathcal{R}^{D_E}$ (D_E=2048 in all our experiments in) the paper). Consistent with the findings , our analysis and experiments show that this normalization improves top-1 accuracy.
- *Projection Network*, $Proj(\cdot)$, which maps r to a vector $z = Proj(r) \in R^{D_P}$. We instantiate $Proj(\cdot)$ as either a multi-layer perceptron with a single hidden layer of size 2048 and output vector of size $D_P = 128$ or just a single linear layer of size $D_P = 128$; we leave to future work the investigation of optimal $Proj(\cdot)$ architectures. We again normalize the output of this network to lie on the unit hypersphere, which enables using an inner product to measure distances in the projection space. As in self-supervised contrastive learning, we discard $Proj(\cdot)$ at the end of contrastive training. As a result, our inference-time models contain exactly the same number of parameters as a cross-entropy model using the same encoder, $Enc(\cdot)$.

Given this framework, we now look at the family of contrastive losses, starting from the selfsupervised domain and analyzing the options for adapting it to the supervised domain, showing that one formulation is superior. Fora set of N randomly sampled sample/label pairs, $\{x_k, y_k\}_{k=1...N}$, the corresponding batch used for training consists of $2N$ pairs, $\{\tilde{x}_\ell, \tilde{y}_\ell\}_{\ell=1...2N}$, where $\tilde{x}_{2k}$ and $\tilde{x}_{2k-1}$ are two random augmentations (a.k.a.. "views") of $x_k\,(k = 1...N)$ and $\tilde{y}_{2k-1} = \tilde{y}_{2k} = y_k$. augmented samples as a "multiviewed batch".

C. Two Stage Training

We divided our training to two phases for each experience. The first place is supervised, contrastive learning. It's a representation learning with a contrastive loss function that does two things.It pulls the embedding of the same class close to each other. And it pushes the embedding of different classes apart The second phase is how to classification. But only in the classes of the current experiences are getting trained.All replay samples are classified as a single out of distribution class, and we trained them as an extra class.

$$\mathcal{L}^{sup} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp\left(\boldsymbol{z}_i \cdot \boldsymbol{z}_p / \tau\right)}{\sum_{a \in A(i)} \exp\left(\boldsymbol{z}_i \cdot \boldsymbol{z}_a / \tau\right)} \quad (3)$$

So, our model given an experience ID can only generate the logits of the classes from that experience.Therefore, during the prediction we have to go through multiple experiences to gather logits of all the classes. Naturally, we believe that the latest experience been trained with more data would give us better logits, but that is not always the case every time.The logits from previous experience were approved to be very helpful.Consequently, we adopt a momentum-based approach that utilizes the last three experiences for every logit of a class. Additionally, since our model has trained with the data orientation. We also apply the test time augmentation.

D. Model With Replicas

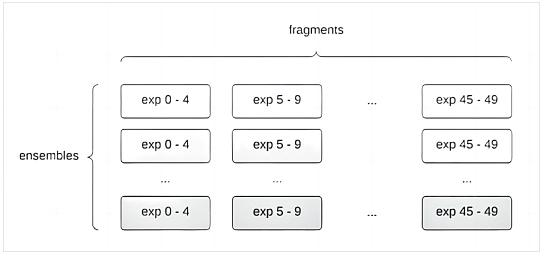


Fig. 3: fragments and ensemble figure

So far, everything we have covered was in a single model with no replicas.Our replica approach was implemented on top of the single model approach. It has two types of verticals. The first is called fragments, which is a technique to break down the whole learning stream. Into the chunks of consecutive experiences. The second type is called Ensemble, ensembles are essentially identical models initialized with different weights.We operate independently,but share the

same computational graph. These two types of replicas are orthogonal to each other, allowing us to have any combination of fragments and ensembles simultaneously.

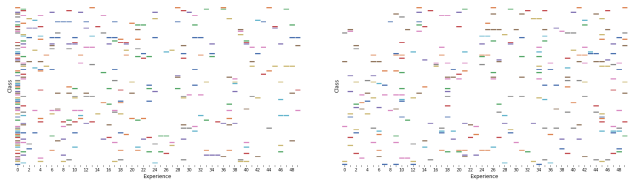
IV. Experience

A. Dataset

Our experimental dataset is CIFAR-100, and the CIR data stream generator is used on the basic dataset to generate corresponding CIR data stream. The generator used for this challenge creates random streams via four interpretable control parameters:

- *Stream Length* (N): number of experiences in the stream.
- *Experience Size* (S): number of patterns in each experience. By default, experience size is equally divided between present classes in each experience.
- *First Occurrence Distribution* (P_f): a discrete probability distribution over the experiences in the stream that determines how the first occurrences of dataset classes can happen throughout the stream.
- *Repetition Probability* (P_r): per-class repetition probabilities that control how likely it is for each class to re-appear after its first occurrence in the stream. In the simplest form, it is a list of probability values, one for each class.

The "first occurrence" control parameter P_r determines the timing of when dataset classes appear for the first time in the stream. For instance, in one stream, all classes may appear for the first time in the beginning, and in another stream, new classes may appear randomly throughout the stream with equal probability. It is important to investigate how changing P_r may affect the models' learning and thus design strategies that are more robust to such changes in the stream.



(a) Most of the classes are observed in the first 5 experiences.

(b) Novel classes can appear throughout the stream.

Fig. 4: some examples of CIR data stream

There are 50 experiences in each stream,and there will be 2000 samples in each experience. Samples are equally divided between present classes in each experience. As shown in Fig 4, the left one's p is 0.6, and 0.01 for right one.And then calculate the $f(i,p) = (1-p)^{i-1}p$.

B. Baselines

Our baselines are set as below:

- Naive: train the network using traditional methods
- Base avalanche [6] plugins: we will use some basic plugins from avalanche, such as LwF, EWC.
- Our Replay plugin: we adopt replay method which is able to store origin data for each class for the dataset stream.
- Champion's Algorithm on single model: this method combines the strengths of Hard Attention to the Task (HAT) and Supervised Contrastive Learning in the context of Class-Incremental with Repetition (CIR)
- Champion's Algorithm with Replicas: Specifically, two strategies—one employing specialized models for different experiences, and the other using ensemble models trained on the same experience

C. Result

The test dataset owns about 10000 images covering all 100 classes, it's able to Assessment the accuracy.The results are as follows:

	naive	base plugin	replay plugin	single model	with replicas
s1	6.63	7.50	37.26	46.17	68.04(avg)
s2	8.89	7.03	33.16	56.63	

TABLE I: Results table about test accuracy, with %

From the experimental results, we can find that Naive's training strategy has poor results. After adding some of Avalanche's training strategy plug-ins, there is no better effect improvement. This may be because these plug-ins are not suitable for the current data flow form. After using our own experience playback plug-in, the effect has been greatly improved, reaching a maximum of 37.26%. Regarding the algorithm results of the winner of the competition, both a single model and a model with a copy have better performance, but the model after adding a copy has nearly twice the training time compared to the original model.

V. Conclusion

In this course task, we first learned about the concept of Class-incremental with Repetition. Compared with the current mainstream class incremental and domain incremental learning, Class-incremental with Repetition learning has received less attention, and there are many directions waiting for our exploration. Secondly, we actually used continuous learning related algorithms through programming and gained a more nuanced understanding of continuous learning. At the same time, in the process of exploring competition challenges, we referred to many excellent algorithms from our predecessors. They often integrated existing algorithms and made targeted improvements based on current tasks. We also learned classic continuous learning algorithms and new algorithm ideas. On this basis, we programmed ourselves to implement a simple experience playback mechanism. Although it is relatively simple, it is also a relatively big progress for us. Finally, we summarized different algorithms and conducted experiments, hoping that these experiences can help our future scientific research work.

Although we have conducted some related experiments, overall the actual amount of code written is relatively small, and we have not fully mastered the Avalanche continuous learning framework. At the same time, we still have some unclear understanding of some of the details of the algorithm that combines HAT and supervised contrastive learning proposed

by the winner of the competition, and we need to continue to work hard in the future. Finally, we hope to add some new modules to the champion's algorithm; or use strategies such as model expansion for training to achieve better results.

VI. Author contributions

1) Shuai Xue:

- Implemented our own experience replay plugin.
- Send an email to the champion to obtain relevant code files and parameter configurations.
- Run part of our experiment.
- Complete part of thesis writing.

2) Yuanzhuo Niu:

- Employing Switching Plugins and reproduce HAT with two replicas from the winner of this competition.
- Read the paper about HAT attentively, and understand its mechanisms.
- Organize group work, and write the PowerPoint for group presentation individually.

References

[1] H. Hemati, A. Cossu, A. Carta, J. Hurtado, L. Pellegrini, D. Bacciu, V. Lomonaco, and D. Borth, "Class-incremental learning with repetition," in *Conference on Lifelong Learning Agents*. PMLR, 2023, pp. 437–455.

[2] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *International conference on machine learning*. PMLR, 2018, pp. 4548–4557.

[3] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[4] 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*, vol. 114, no. 13, pp. 3521–3526, 2017.

[5] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *Advances in neural information processing systems*, vol. 33, pp. 18 661–18 673, 2020.

[6] V. Lomonaco, L. Pellegrini, A. Cossu, A. Carta, G. Graffieti, T. L. Hayes, M. De Lange, M. Masana, J. Pomponi, G. M. Van de Ven *et al.*, "Avalanche: an end-to-end library for continual learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3600–3610.